İSTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

RELIABILITY BASED STRUCTURAL and AEROELASTIC OPTIMIZATION of WING MODELS with HIGH FIDELITY SOLVERS

M.Sc. Thesis by
Muhammet Nasıf KURU

Department : Informatics

Programme : Computational Science and Engineering

APRIL 2011

İSTANBUL TECHNICAL UNIVERSITY ★ INFORMATICS INSTITUTE

RELIABILITY BASED STRUCTURAL and AEROELASTIC OPTIMIZATION of WING MODELS with HIGH FIDELITY SOLVERS

M.Sc. Thesis by
Muhammet Nasıf KURU
(702071011)

Date of submission : 20 March 2011
Date of defence examination: 28 April 2011

Supervisor (Chairman) : Assoc. Prof. Dr. Melike NİKBAY (ITU)
Members of the Examining Committee : Asst. Prof. Dr. Erdem ACAR (TOBB ETU)
Asst. Prof. Dr. Murat MANGUOĞLU (ODTU)

APRIL 2011

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ BİLİŞİM ENSTİTÜSÜ

KANAT MODELLERİNİN YÜKSEK DOĞRULUKLU ÇÖZÜCÜLERLE GÜVENİLİRLİK TABANLI YAPISAL ve AEROELASTİK ENİYİLEŞTİRMESİ

YÜKSEK LİSANS TEZİ

Muhammet Nasıf KURU
(702071011)

Tezin Enstitüye Verildiği Tarih : 20 Mart 2011
Tezin Savunulduğu Tarih : 28 Nisan 2011

Tez Danışmanı : Doç. Dr. Melike NİKBAY (İTÜ)
Diğer Jüri Üyeleri : Yrd. Doç. Dr. Erdem ACAR (TOBB ETÜ)
Yrd. Doç. Dr. Murat MANGUOĞLU (ODTÜ)

NİSAN 2011

FOREWORD

I would first like to thank my family very much who are the primary people behind my every achievement. I would also like to thank my advisor, Assoc. Prof. Dr. Melike Nikbay for her support and guidance during this work. I acknowledge Asst. Prof. Dr. Erdem Acar's support and valuable suggestions.

Many thanks and much appreciation to my dear friends, Necati Fakkusoğlu, Semra Bayat, Fırat Gür, Süha Öz and Emre Tanır.

April 2011

Muhammet Nasıf KURU

Mechanical Engineer

TABLE OF CONTENTS

Page

ABBREVIATIONS

RBDO	**:** Reliability Based Design Optimization
RIA	**:** Reliability Index Approach
PMA	**:** Performance Measure Approach
PDF	**:** Probability Density Function
CDF	**:** Cumulative Distribution Function
JPDF	**:** Joint Probability Density Function
MVFOSM	**:** Mean Value First Order Second Moment
FORM	**:** First Order Reliability Method
SORM	**:** Second Order Reliability Method
MPP	**:** Most Probable failure Point
HL-RF	**:** Hasofer Lind - Rackwitz Fiessler
HL	**:** Hasofer Lind
AMV	**:** Advanced Mean Value
SF	**:** Safety Factor
AGARD	**:** Advisory Group for Aerospace Research and Development

LIST OF SYMBOLS

$\boldsymbol{b}$: vector of deterministic design variables; $\boldsymbol{b} = [b_1, b_2, \dots, b_n]$

$\boldsymbol{X}$: random parameters or variables of the system; $\boldsymbol{X} = [X_1, X_2, \dots, X_n]^T$

$\boldsymbol{x}$: $\boldsymbol{X}$'s particular value; $\boldsymbol{x} = [x_1, x_2, \dots, x_n]^T$

$\boldsymbol{x}^*$: most probable failure point (MPP) in X-space

$\boldsymbol{U}$: independent and standard normal random parameters or variables of the system; $\boldsymbol{U} = [U_1, U_2, \dots, U_n]^T$

$\boldsymbol{u}$: $\boldsymbol{U}$'s particular value; $\boldsymbol{u} = [u_1, u_2, \dots, u_n]^T$

$\boldsymbol{u}^*$: most probable failure point (MPP) in U-space

$f_{X_i}(x_i)$: probability density function (PDF) of *i*th random parameter X_i

$F_{X_i}(x_i)$: cumulative distribution function (CDF) of *i*th random parameter X_i

$f_{\boldsymbol{X}}(\boldsymbol{x})$: joint probability density function (JPDF) of $\boldsymbol{X}$

μ_{X_i} : mean value of *i*th random parameter X_i

$E[.]$: expectation operator

$V[.]$: variance operator

$V[X_i]$: variance of *i*th random parameter X_i

σ_{X_i} : standard deviation of *i*th random parameter X_i

Ξ_i : standard normally distributed variable

ξ_i : particular value of Ξ_i

$\phi(.)$: standard normal probability density function

$\Phi(.)$: standard normal cumulative density function

$\Phi^{-1}[.]$: inverse of the standard normal cumulative distribution function $\Phi[.]$

$g(.)$: limit-state or performance function

$g(.) = 0$: limit-state or failure surface

P_f : probability of failure

$P(.)$: probability function

P_R : required probability of failure level of the system

β_s : safety or reliability index

β_t : target safety or reliability index

g_p : probabilistic performance measure or performance measure

α : sensitivity factor

$\boldsymbol{n}$: normalized steepest descent direction of performance function

LIST OF TABLES

Page

LIST OF FIGURES

Page

RELIABILITY BASED STRUCTURAL and AEROELASTIC OPTIMIZATION of WING MODELS with HIGH FIDELITY SOLVERS

SUMMARY

In engineering design, uncertainties related to geometries, material properties, manufacturing processes and operating conditions are inevitable factors which should be accurately quantified and included while designing and optimizing a realistic system for a required level of reliability and efficiency.

In this thesis, reliability based design optimization (RBDO) methodology is constructed by coupling high-fidelity commercial solvers for aeroelastic analysis and an in-house code developed for reliability analysis.

A RBDO benchmark problem (from the literature) and the developed methodology is validated. An in-house code is integrated to commercial software for aircraft wing applications. Finally the methodology is applied to a fluid-structure interaction (FSI) problem where reliability based structural optimization of a simple aircraft wing and reliability based aeroelastic optimization of AGARD 445.6 wing are performed.

In the final application, the optimization criteria include both deterministic and probabilistic constraints with both structural and aerodynamic uncertainties such as in yield strength, Mach number and angle of attack. To evaluate the probability of failure for the probabilistic constraints, first order reliability analysis methods, Hasofer-Lind (HL) iteration method and advanced mean value (AMV) method are implemented in Matlab to compute most probable failure point (MPP) solution.

KANAT MODELLERİNİN YÜKSEK DOĞRULUKLU ÇÖZÜCÜLERLE GÜVENİLİRLİK TABANLI YAPISAL ve AEROELASTİK ENİYİLEŞTİRMESİ

ÖZET

Mühendislik tasarımında, geometriye, malzeme özelliklerine, üretim süreçlerine ve işletim koşullarına bağlı belirsizlikler kaçınılmazdır. Bu belirsizlikler doğru olarak değerlendirilmeli ve sistemler tasarlanırken ve eniyilenirken hesaba katılmalıdırlar.

Bu çalışmada, güvenilirlik tabanlı tasarım eniyileme (GTTE) metodolojisi oluşturulmuştur. Burada hem ticari mühendislik yazılımları hem de kendimizin geliştirilmiş güvenilirlik kodu ilk uygulama olarak, literatürden alınan ankastre kiriş örneğiyle doğrulanmıştır. Daha sonra genel kanat yapısının eniyilemesi ve en son olarak AGARD 445.6 kanadının aeroelastik eniyilemesi problemine uygulanmıştır.

Ele alınan en son problemde, eniyileme kriterleri arasında akma mukavemeti, Mach sayısı ve hücum açısı gibi yapısal ve aerodinamik parametrelere ait belirsizlikler, olasılıksal kısıtlamalarda kullanılmıştır, ayrıca deterministik kısıtlamalarda mevcuttur. Olasılıksal kısıtların hata olasılığını hesaplamak için, birinci dereceden güvenilirlik analiz metodlarından olan Hasofer-Lind iterasyon metodu ve geliştirilmiş ortalama değer (GOD) metodu Matlab'da uygulanmıştır. Böylece en olası hasar noktası (EON) çözümü hesap edilmiştir.

1. INTRODUCTION

1.1 Background and Literature Review of Reliability Based Design Optimization

Today, in aircraft industry, there is a great competition to release new aircraft designs which are faster, more efficient, more economical, more reliable and even quieter than the former ones both in military and civil applications. The challenging multi-disciplinary task of aircraft design can be realized by incorporation of numerical optimization techniques in the industrial design process. However, there are always uncertainties related to design parameters, modelling, manufacturing process, operating conditions and human factors when designing a new aircraft. Aerospace structures have been designed traditionally by using deterministic approaches based on Federal Aviation Administration (FAA) regulations for a high level of safety. However, it is known that deterministic optimization techniques may lead to unreliable or inefficient designs since they can not consider the uncertainties in different design parameters simultaneously.

In engineering design, uncertainties related to geometries, material properties, manufacturing processes and operating conditions are inevitable factors which should be accurately quantified and included while designing and optimizing a realistic system for a required level of robustness and efficiency. For that reason, recently there is a growing interest in replacing deterministic design approaches with uncertainty-based stochastic computations to produce more robust and efficient structures. In reliability-based stochastic computations, uncertainty can be represented using random variables, processes, and fields.

In general, reliability analysis methods can be categorized into three main types: sampling methods, projection methods, and moment methods. The application of these methods is usually based on their accuracy, computational costs, ease of implementation, and the area interest in the response distribution (mean or tails).

Sampling methods, such as the Monte Carlo Simulation method [1,2], are widely used due to their generality, simplicity, and effectiveness on problems that are highly nonlinear with respect to the uncertainty parameters. A basic advantage of sampling methods is their direct utilization of experiments to obtain mathematical solutions or probabilistic information concerning problems whose system equations cannot be solved easily by known procedures. However, as practical engineering applications typically require a high level of reliability, a large number of samples is needed to obtain accurate results. Therefore, Monte Carlo simulation is impractical for implicit systems solved by high-fidelity numerical simulation. Improved sampling methods, such as importance sampling [3,4], adaptive importance sampling [5,6], and radial importance sampling [7], reduce the number of samples by up to 20 times in comparison with Monte Carlo simulation [8], but are still computationally too expensive, for example, for the use with high-fidelity aeroelastic simulations.

Stochastic projection methods are based on an explicit expansion of the systems' response, such as the polynomial chaos expansion (PCE) [9] and Karhunen-Loeve (KL) expansion. The size and order of the expansion depends on the nonlinearity of the system with respect to the input randomness and the number of random inputs. The computational cost increases significantly with the order of approximation and the number of random inputs. The application of stochastic projection methods to structures is well explored, and mature computational procedures have been developed [10,11]. Stochastic flow problems have also been frequently studied in the literature [12,13], not including the vast amount of literature on stochastic modeling of turbulence. However, the application of stochastic projection methods to fluid structure interaction (FSI) problems is still in its infancy. For characterizing and quantifying stochastic variations around the mean value, a polynomial chaos expansion has been applied to the aeroelastic state equations for small academic problems by Xiu et al. [14].

Moment methods approximate the limit state of an event in question to simplify the integration of the response probability density function over the area of occurrence. The mean value first-order second moment (FOSM) method [15] is frequently used to approximate the influence of random input on the stochastic system response. However, as the mean value point is usually not found on the failure surface, the FOSM approach typically leads to inaccurate results and the prediction is sensitive to

the mathematical formulation of the limit state function [16]. First- and Second-Order Reliability Methods (FORM and SORM) employ an approximation of the limit state function at the Most Probable Point (MPP) of failure [17]. FORM requires the first-order derivatives to linearize the failure function at the MPP, and therefore it is considered accurate as long as the curvature of the failure function in the space of the uncertainty variables is not too large at the MPP. SORM approximates highly nonlinear systems more accurately than FORM, but requires the first and second order derivatives to build a quadratic approximation of the failure surface at the MPP.

Reliability is the probability that a system will perform its function over a specified period of time and under specified service conditions. Assesing reliability within a design optimization context is broadly useful, and reliability based design optimization (RBDO) methods are popular approaches for designing systems while accounting for uncertainty [18]. In RBDO, the statistical nature of constraints and design problems are defined in the objective function and probabilistic constraint. In RBDO, the cost is optimized subject to prescribed probabilistic constraints by solving a mathematical nonlinear programming problem. Therefore, the solution from RBDO provides not only an improved design but also a higher level of confidence in the design [19].

In general, an RBDO model includes deterministic design variables, random design variables and random parameters. A deterministic design variable is a design variable to be designed with its negligible uncertainties. A random design variable is a variable to be designed with uncertainty property being considered (usually the mean of the variable is to be determined) while a random parameter can not be controlled.

The probability distributions can be used to describe the stochastic nature of the random design variables and random parameters, where the variations are represented by standard deviations which are usually assumed to be constant. Thus, a typical RBDO problem can be defined as a stochastic optimization model with the objective function over the mean values of design variables (deterministic and stochastic) is to be optimized, subject to probabilistic constraints.

Two essential components of RBDO are reliability analysis and optimization. Reliability analysis focuses on analyzing the probabilistic constraints to ensure the

reliability levels are satisfied while optimization is seeking for the minimum objective function subjected to the probabilistic constraints.

Reliability analysis perform uncertainty quantification (UQ) by computing approximate response function distribution statistics based on specified input random variable probability distributions. Extensive research has been done to explore various efficient reliability analysis techniques including sensitivity-based approximation approaches by Eggert [20], Parkinson [21], MPP (most probable failure point) based approaches by Hohenbichler [22], Koyluoglu [23], Hasofer [17], Monte Carlo Simulations (MCS) and Response Surface Model based approaches by Chen [24], Sues [25], Koch [26]. Among those, MPP based approaches have attracted more attention as they require relatively less computational effort while still producing results with acceptable accuracy compared to the other three approaches [27,28].

Reliability based design optimization (RBDO) involves evaluation of probabilistic constraints, which can be done in two different ways, the reliability index approach (RIA) and the performance measure approach (PMA). Popular numerical methods for RIA are the Hasofer Lind-Rackwitz Fiessler (HL-RF) method [17,29], modified HL-RF [29], and two-point approximation [30,31]. For PMA, the Advanced Mean Value (AMV) method [5,32] is a popular numerical method.

Another research topic in RBDO is on integration of reliability analysis and optimization, using bi-level strategy or sequential strategy. The resulting RIA/PMA algorithms can be employed within bi-level or sequential RBDO approaches.

Bi-level methods (Figure 1.1) treat the reliability analysis as the inner loop analyzing the probabilistic constraint satisfaction for the given solutions provided by the outer optimizer which locates the optimal solution iteratively. As a result, bi-level methods are computationally expensive for a complex engineering design [27,33,34]. Therefore, sequential methods have been developed to address the computational challenges as in the work of Zou and Mahadevan [27], Du and Chen [34], Thanedar [35], Tu [36], Chen [37], Royset [38] and Youn [39].

Integration of RBDO methodologies to aerospace engineering applications has been a challenging research subject recently. Petit [41] presented general sources of uncertainty on aeroelastic response such as flutter flight testing, prediction of limit

cycle oscillations and design optimization with aeroelastic constraints and reviewed research challenges in this field.

Allen and Maute [42] presented a computational methodology that both utilizes high-fidelity simulation methods and accounts for uncertainties in design and operating conditions within the design process of aeroelastic structures.

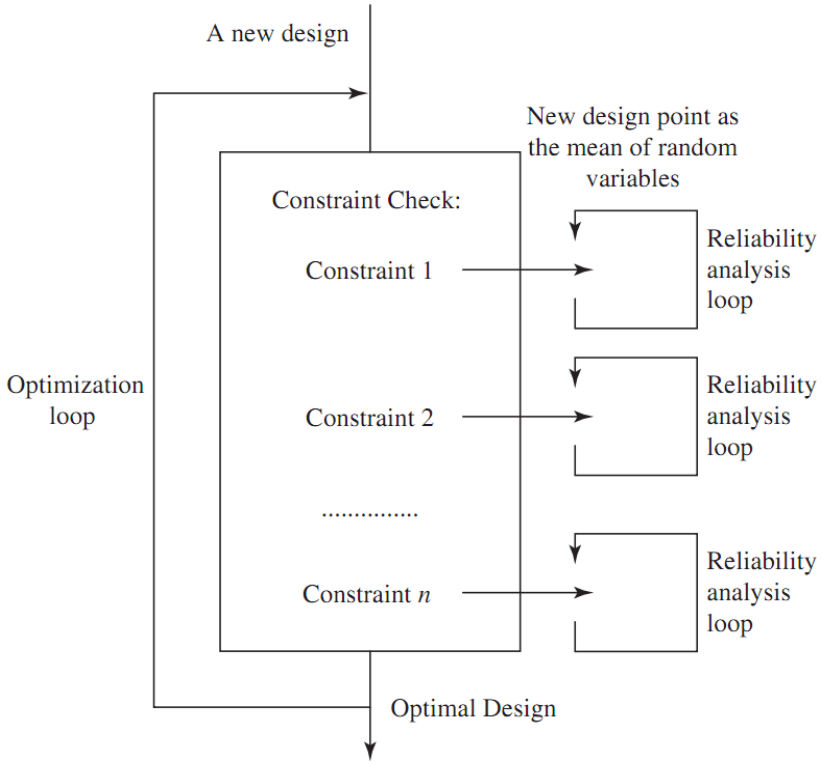


Figure 1.1 : Flowchart of bi-level reliability based design optimization (RBDO) [40]

Hosder et al. [43] presented an inexpensive non-intrusive polynomial chaos (NIPC) method for the propagation of input uncertainty in CFD simulations. Moreover, this NIPC approach has been applied to three different problems which were an inviscid oblique shock wave problem with geometric uncertainty, an inviscid expansion wave problem with geometric uncertainty, and a subsonic, two dimensional, laminar boundary layer flow over a flat plate with an uncertain free-stream dynamic viscosity.

Kwon et al. [44] performed a reliability analysis for the aerodynamic analysis of a 2D airfoil, a 3D wing and a wing body configuration by using moment method. A stochastic spectral projection solver based on generalized polynomial chaos expansion was applied to the uncertainty quantification of stochastic compressible

flows around a NACA0012 airfoil due to random free-stream Mach number and angle of attack by Chassaing et al. [45].

Recently, Missoum et. al. [46] presented a methodology which constructed explicit flutter and subcritical limit cycle oscillation boundaries in terms of deterministic and random design variables for the reliability-based design optimization of systems with nonlinear aeroelastic constraints.

In this thesis, firstly, a deep investigation of RBDO techniques is presented and then a fully automatic, modular and practical design framework which employs RBDO techniques within a multidisciplinary code coupling approach based on high-fidelity CAD, CFD and CSD softwares and fluid-structure interface is developed and applied to aeroelastic optimization problems. In this computational framework, finite volume based flow solver Fluent is used to solve inviscid 3D Euler equations and Catia is used as a parametric 3D solid modeler. Abaqus, a structural finite element method solver, is used to compute the structural response of the aeroelastic system. Mpcci, mesh based parallel code coupling interface, is used to exchange the pressure and displacement information between Fluent and Abaqus to perform a loosely coupled aeroelastic analysis. Modefrontier is employed as a multi-objective and multi-disciplinary optimization driver to control the optimization workflow.

The optimization criteria include both deterministic and probabilistic constraints with both structural and aerodynamic uncertainties such as in yield strength, Mach number and angle of attack. To evaluate the probability of failure for the probabilistic constraints, first order reliability method (FORM) is used. In RIA, Hasofer-Lind iteration method is implemented in Matlab to compute MPP (Most Probable failure Point) solution. In PMA, AMV method is implemented in Matlab to compute MPP solution. The integrated framework is validated with academic and structural problems and then extended to more realistic wing configurations with aeroelastic criteria.

Deterministic optimization studies with multidisciplinary code coupling approach were presented in the former work of Nikbay et al. [47,48]. Present work is an RBDO extension of the former MDO frameworks in Ref. [47] and [48] so that day-to-day codes can be still used in an attach/detach manner for realistic problems with stochastic nature.

1.2 Purpose and Outline of the Thesis

The main purpose of this work is to learn and take advantage of the reliability based design optimization concept and underline its importance for the practical industrial applications with uncertainty parameters.

In the second chapter, reliability based design optimization is introduced. Mathematical approaches about reliability analysis and inverse reliability analysis are given and the related methods are presented.

The third chapter covers the verification of the implemented algorithm. A benchmark problem with a cantilever beam design from the literature is solved and the methodology is validated. Different reliability analysis methods are compared in terms of efficiency.

The fourth chapter includes the integration of the written code and commercial softwares for the optimization problems presented formerly by Nikbay et al. [47,48]. Reliability based structural optimization of a simple aircraft wing and reliability based aeroelastic optimization of AGARD 445.6 wing are performed.

In the fifth chapter, conclusions are drawn based on the results.

2. RELIABILITY BASED DESIGN OPTIMIZATION

Assesing reliability within a design optimization context is broadly useful, and reliability based design optimization (RBDO) methods are popular approaches for designing systems while accounting for uncertainty [18]. In RBDO, the statistical nature of constraints and design problems are defined in the objective function and probabilistic constraint.

RBDO involves evaluation of probabilistic constraints, which is usually done in two different ways, the reliability index approach (RIA) and the performance measure approach (PMA). The resulting RIA/PMA algorithms can be employed within bi-level or sequential RBDO approaches. In this study, bi-level RBDO approach is used.

2.1 Design Optimization Problem Formulation

Standard design optimization model can be formulated as:

$$Minimize \ : \ f(\boldsymbol{b}) = f(b_1, b_2, \dots, b_n)$$

$$Subject\ to: \ g_i(\boldsymbol{b}) = g_i(b_1, b_2, \dots, b_n) \leq 0, \quad i = 1, \dots, n_g$$

$$h_j(\boldsymbol{b}) = h_j(b_1, b_2, \dots, b_n) = 0, \quad j = 1, \dots, n_h$$

$$\boldsymbol{B} = \{\boldsymbol{b} \in \mathbb{R}^n | \boldsymbol{b_L} \leq \boldsymbol{b} \leq \boldsymbol{b_U}\} \tag{2.1}$$

where $\boldsymbol{B}$ is a set of design variables restricted by lower and upper bounds $\boldsymbol{b}_L$ and $\boldsymbol{b}_U$, $f(\boldsymbol{b})$ is cost function, $g_i(\boldsymbol{b})$ is a set of inequality constraints, $h_j(\boldsymbol{b})$ denotes a set of equality constraints, and n, n_g, n_h are the number of design variables, inequality and equality constraints respectively.

2.2 Structural Reliability

2.2.1 Basic Probabilistic Descriptions

Random Variable : The uncertainties of an engineering system are identified by the variations of the random vector $\boldsymbol{X} = [X_1, X_2, \dots, X_n]^T$**,** which can be random design

variables or random parameters of the system. $\boldsymbol{X}$'s particular value is represented by $\boldsymbol{x} = [x_1, x_2, \dots, x_n]^T$.

Probability Density Function : The probability distribution of X_i is described by its probability density function (PDF) $f_{X_i}(x_i)$. $f_{X_i}(x_i)$ is the PDF of X_i as shown in Figure 2.1.

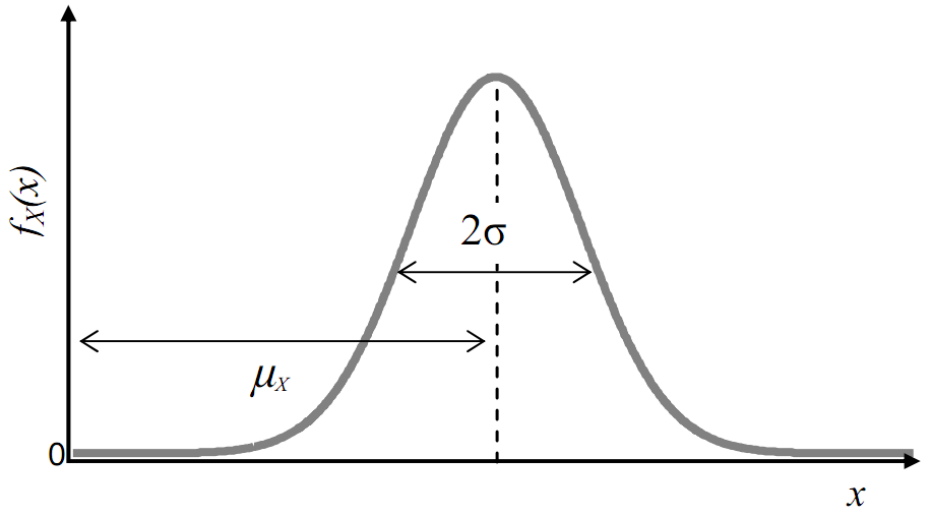


Figure 2.1 : The probability density function (PDF) of normal distribution [50]

Cumulative Distribution Function : The cumulative distribution function (CDF) $F_{X_i}(x_i)$ describes the probability that a random variable X_i with a given probability distribution will be found at a value less than or equal to x_i. For every real number x_i, the CDF of a random variable X_i is given by

$$F_{X_i}(x_i) = P(X_i \leq x_i) \tag{2.2}$$

where the right-hand side represents the probability that the random variable X_i takes on a value less than or equal to x_i. $F_{X_i}(x_i)$ is the CDF of X_i as shown in Figure 2.2.

Joint Probability Density Function : The probability function that two or more random events will happen simultaneously (JPDF) $f_{\boldsymbol{X}}(\boldsymbol{x})$. For instance, the probability of the two-dimensional case is calculated as

$$P[a < X < b\,, c < Y < d] = \int_c^d \int_a^b f_{XY}(x, y)\, dx\, dy \tag{2.3}$$

where $f_{XY}(x, y)$ is the joint PDF (JPDF) of the random variables X and Y.

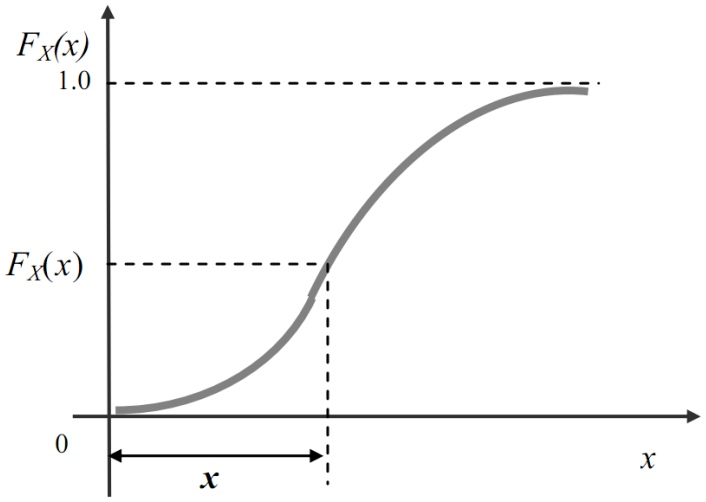


Figure 2.2 : The cumulative distribution function (CDF) of normal distribution [50]

Mean : This is a weighted average of all the values that a random variable may take. If $f_{X_i}(x_i)$ is the PDF of X_i, the mean is given by

$$\mu_{X_i} = E[X_i] = \int_{-\infty}^{\infty} x_i \, f_{X_i}(x_i) dx_i \quad \textbf{(2.4)}$$

μ_{X_i} is also called the first raw moment. The operator, $E[.]$ denotes the average or expected value of X_i, possesses the following useful properties: If X_1 and X_2 are independent random parameters,

$$E[X_1 X_2] = E[X_1] \, E[X_2] \quad \textbf{(2.5)}$$

$$E[X_1 + X_2] = E[X_1] + E[X_2] \quad \textbf{(2.6)}$$

and if c is a constant,

$$E[c] = c \quad \textbf{(2.7)}$$

$$E[cX_i] = c \, E[X_i] \quad \textbf{(2.8)}$$

Variance and Standard Deviation : The variance, $V[X_i]$, a second central moment of X_i, is a measure of spread in the data about mean:

$$V[X_i] = E\left[\left(X_i - \mu_{X_i}\right)^2\right] \quad \textbf{(2.9)}$$

The standard deviation is a square root of the variance:

$$\sigma_{X_i} = \sqrt{V[X_i]} \quad \textbf{(2.10)}$$

The variance operator, $V[.]$, possesses the following useful properties: If X_1 and X_2 are independent random parameters,

$$V[X_1 + X_2] = V[X_1] + V[X_2] \quad \textbf{(2.11)}$$

and if c is a constant,

$$V[c] = 0 \quad \textbf{(2.12)}$$

$$V[cX_i] = c^2\, V[X_i] \quad \textbf{(2.13)}$$

Normal (Gaussian) Distribution : The normal distribution is given by

$$f_{X_i}(x_i) = \frac{1}{\sigma_{X_i}\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{x_i-\mu_{X_i}}{\sigma_{X_i}}\right)^2\right] \;, \quad -\infty < x_i < \infty \quad \textbf{(2.14)}$$

where μ_{X_i} and σ_{X_i} denote the mean and standard deviation of the variable X_i, respectively, and X_i is identified as $N(\mu_{X_i}, \sigma_{X_i})$ if it has a normal (gaussian) distribution.

The gaussian distribution can be normalized by defining

$$\xi_i = \frac{x_i-\mu_{X_i}}{\sigma_{X_i}} \quad \textbf{(2.15)}$$

and yields the standard normal distribution $N(0,1)$ with zero mean and unit standard deviation.

The PDF of the standard normally distributed variable ξ_i is given by

$$\phi(\xi_i) = f_{\Xi_i}(\xi_i) = \frac{1}{\sqrt{2\pi}} exp\left[-\frac{1}{2}{\xi_i}^2\right] \;, \quad -\infty < \xi_i < \infty \quad \textbf{(2.16)}$$

where $\phi(.)$ represents the standard normal probability density function.

The CDF of the standard normally distributed variable ξ_i is given by

$$\Phi(\xi_i) = F_{\Xi_i}(\xi_i) = \int_{-\infty}^{\xi_i} \frac{1}{\sqrt{2\pi}} exp\left[-\frac{1}{2}{\xi_i}^2\right] d\xi_i \quad \textbf{(2.17)}$$

where $\Phi(.)$ represents the standard normal cumulative distribution function. The values of the standard normal cumulative distribution function, $\Phi(.)$, are tabulated in Appendix A.1 [50].

2.2.2 Structural Reliability Assessment

When a structure exceeds a specific limit in one of its design criteria, the structure fails to perform as it is required. For example, exceeding yield strength of a material may cause plastic deformation of the structure. This specific limit is called a limit-state. If the probability of failure of the structure exceeds the required value, the structure will be considered unreliable.

The limit-state function or performance function, $g(.)$ and probability of failure, P_f, can be defined as

$$g(\boldsymbol{X}) = R(\boldsymbol{X}) - S(\boldsymbol{X}) \quad \textbf{(2.18)}$$

$$P_f = P[g(.) < 0] \quad \textbf{(2.19)}$$

where R is the resistance and S is the actual loading of the system. Both $R(.)$ and $S(.)$ are functions of random variables $\boldsymbol{X}$. The notation $g(.) < 0$ inequality denotes the failure region, $g(.) = 0$ and $g(.) > 0$ indicate the failure surface and safe region, respectively.

The mean and standard deviation of the limit-state, $g(.)$, can be determined from the elementary definition of the mean and variance (Equations (2.6) and (2.11)). The mean of $g(.)$ is

$$\mu_g = \mu_R - \mu_S \quad \textbf{(2.20)}$$

where μ_R and μ_S are the means of R and S, respectively. If R and S are independent, the standard deviation of $g(.)$ is

$$\sigma_g = \sqrt{\sigma_R^2 + \sigma_S^2} \quad \textbf{(2.21)}$$

where σ_R and σ_S are the standard deviations of R and S, respectively.

Reliability index, β_s, is defined as;

$$\beta_s = \frac{\mu_g}{\sigma_g} = \frac{\mu_R - \mu_S}{\sqrt{\sigma_R^2 + \sigma_S^2}} \quad \textbf{(2.22)}$$

Figure 2.3 shows a geometrical illustration of the reliability index in a one-dimensional case.

The idea behind the reliability index is that the distance from point μ_g to the limit-state surface provides a good measure of reliability. The distance is measured in units

of the uncertainty scale parameter σ_g. The shaded area of Figure 2.3 identifies the probability of failure.

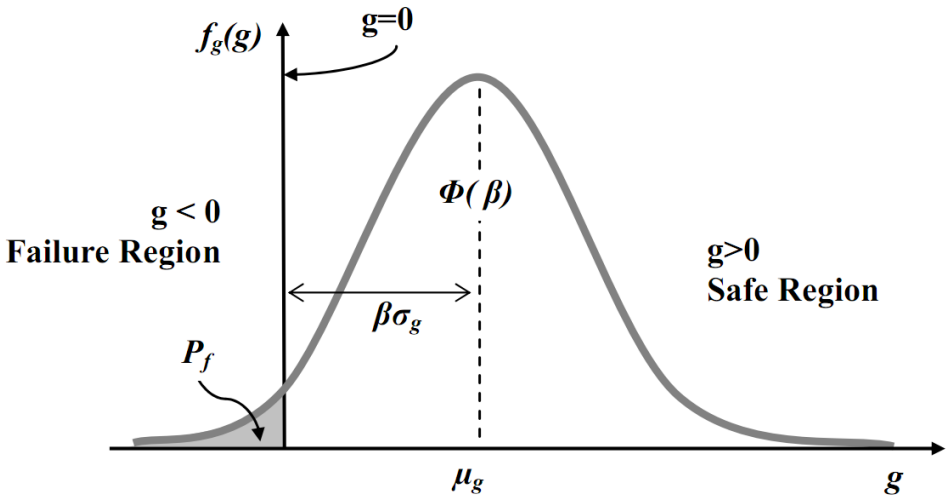


Figure 2.3 : Probability density function (PDF) for limit-state function $g(.)$ [50]

For a special case, the resistance, *R*, and actual loading, *S*, are assumed to be normally distributed and independent. The limit-state function is also normally distributed, since $g(.)$ is a linear function of *R* and *S*.

Thus, the PDF of the limit-state function is

$$f_g(g) = \frac{1}{\sigma_g\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{g-\mu_g}{\sigma_g}\right)^2\right] , \quad -\infty < g < \infty \tag{2.23}$$

The probability of failure is

$$P_f = P[g < 0] = \int_{-\infty}^{0} f_g(g)\, dg = \int_{-\infty}^{0} \frac{1}{\sigma_g\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{g-\mu_g}{\sigma_g}\right)^2\right] dg \tag{2.24}$$

When $g = 0$, the probability of failure is computed as

$$P_f = \int_{-\infty}^{0} \frac{1}{\sigma_g\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{0-\mu_g}{\sigma_g}\right)^2\right] dg \tag{2.25}$$

$$P_f = \int_{-\infty}^{0} \frac{1}{\sigma_g\sqrt{2\pi}} exp\left[-\frac{1}{2}\beta_s^2\right] dg \tag{2.26}$$

Here, $\beta_s = \frac{g-\mu_g}{\sigma_g}$ so $d\beta_s = \frac{dg}{\sigma_g}$, boundaries are;

when $g = 0 \rightarrow \beta_s = \frac{0-\mu_g}{\sigma_g} = -\beta_s$

when $g = -\infty \rightarrow \beta_s = \frac{-\infty - \mu_g}{\sigma_g} = -\infty$

Equation (2.26) can be rewritten as

$$P_f = \int_{-\infty}^{-\beta_s} \frac{1}{\sqrt{2\pi}} exp\left[-\frac{1}{2}\beta_s^2\right] d\beta_s \quad \textbf{(2.27)}$$

From Equations (2.17) and (2.27), the probability of failure can be obtained as

$$P_f = \Phi(-\beta_s) = 1 - \Phi(\beta_s) \quad \textbf{(2.28)}$$

2.3 General Definition of RBDO Model

When uncertainties are included in a design optimization problem; a RBDO model is formulated as below;

$$Minimize \ : f(\boldsymbol{b}, \boldsymbol{\mu_X})$$

$$Subject\ to: \ P_{fi} = P_i[g_i(\boldsymbol{b}, \boldsymbol{X}) < 0] \leq P_{Ri}, \quad i = 1, \dots, m$$

$$h_j^{det}(\boldsymbol{b}) < 0, \qquad j = 1, \dots, n \quad \textbf{(2.29)}$$

where $g_i(.)$ represents the performance function, $h_j^{det}(.)$ represents a set of deterministic constraints, $\boldsymbol{b}$ is the vector of deterministic design variables, and $\boldsymbol{X}$ is the random parameters of the system. P_{fi} is the probability of failure of the ith probabilistic constraint. P_i is the probability function of the ith probabilistic constraint. P_{Ri} is the required probability of failure level of the ith probabilistic constraint. m is the number of probabilistic constraints. n is the number of deterministic constraints.

The probabilistic constraints are described by the performance functions $g_i(\boldsymbol{b}, \boldsymbol{X})$, their probabilistic models, and their required probability of failure levels P_{Ri}.

Consider a performance function $g_i(\boldsymbol{b}, \boldsymbol{X})$, where $g_i(\boldsymbol{b}, \boldsymbol{X}) < 0$ denotes the failure region, $g_i(\boldsymbol{b}, \boldsymbol{X}) = 0$ and $g_i(\boldsymbol{b}, \boldsymbol{X}) > 0$ indicate the failure surface and safe region, respectively. The statistical description of the failure of the performance function $g_i(\boldsymbol{b}, \boldsymbol{X})$ is characterized by its CDF $F_{g_i}(0)$ as

$$F_{g_i}(0) = P_i[g_i(\boldsymbol{b}, \boldsymbol{X}) < 0] = \int_{g_i(\boldsymbol{b},\boldsymbol{X})<0} \dots \int f_{\boldsymbol{X}}(\boldsymbol{x}) dx_1 \dots dx_n \ , \quad i = 1, \dots, m \quad \textbf{(2.30)}$$

The evaluation of Equation (2.30) requires reliability analysis where the multiple integration is involved as shown in Equation (2.30). Some approximate probability

integration methods have been developed to provide efficient solutions, such as the first order reliability method (FORM) or second order reliability method (SORM). FORM often provides adequate accuracy and is widely used for RBDO applications [39].

Each required probability of failure level of the system P_{Ri} is often represented by the target reliability index as $\beta_{t_i} = -\Phi^{-1}(P_{Ri})$ using Equation (2.28). Hence, any probabilistic constraint in Equation (2.29) can be rewritten using Equation (2.30) as

$$F_{g_i}(0) \leq \Phi(-\beta_{t_i}) \quad \textbf{(2.31)}$$

which can also be expressed in two ways through inverse transformations [51] as

$$\beta_{s_i} = -\Phi^{-1}(F_{g_i}(0)) \geq \beta_{t_i} \quad \textbf{(2.32)}$$

$$g_{p_i} = F_{g_i}^{-1}\big(\Phi(-\beta_{t_i})\big) \geq 0 \quad \textbf{(2.33)}$$

where β_{s_i} and g_{p_i} are respectively called the reliability index and the probabilistic performance measure for the *i*th probabilistic constraint.

Equation (2.32) is employed to describe the probabilistic constraint in Equation (2.29) using the reliability index and it is called the reliability index approach (RIA). At a given design, the evaluation of reliability index β_{s_i} for RIA is performed using reliability analysis. Equation (2.29) can be rewritten using RIA as

$$\begin{aligned} &Minimize \ : f(\boldsymbol{b}, \boldsymbol{\mu_X}) \\ &Subject\ to: \ \beta_{s_i} \geq \beta_{t_i}, \qquad i = 1, \dots, m \\ &\qquad\qquad h_j^{det}(\boldsymbol{b}) < 0, \quad j = 1, \dots, n \end{aligned} \quad \textbf{(2.34)}$$

Equation (2.33) is employed to describe the probabilistic constraint in Equation (2.29) using the probabilistic performance measure and it is called the performance measure approach (PMA). At a given design, the evaluation of probabilistic performance measure g_{p_i} for PMA is performed using inverse reliability analysis. Equation (2.29) can be rewritten using PMA as

$$\begin{aligned} &Minimize \ : f(\boldsymbol{b}, \boldsymbol{\mu_X}) \\ &Subject\ to: \ g_{p_i} \geq 0, \qquad i = 1, \dots, m \\ &\qquad\qquad h_j^{det}(\boldsymbol{b}) < 0, \quad j = 1, \dots, n \end{aligned} \quad \textbf{(2.35)}$$

Transformation of the Random Variables from X-space to U-space

Standardize the random variables;

$$u_i = \frac{x_i - \mu_{X_i}}{\sigma_{X_i}} \quad \textbf{(2.36)}$$

where μ_{X_i} and σ_{X_i} represent the mean and the standard deviation of random variable X_i, respectively. The mean and standard deviation of the standard normally distributed variable, u_i, are zero and unity, respectively.

From Equation (2.36), the mean value point μ_{X_i} in the original space (X-space) is mapped into the origin of the normal space (U-space) as shown in Figure 2.4.

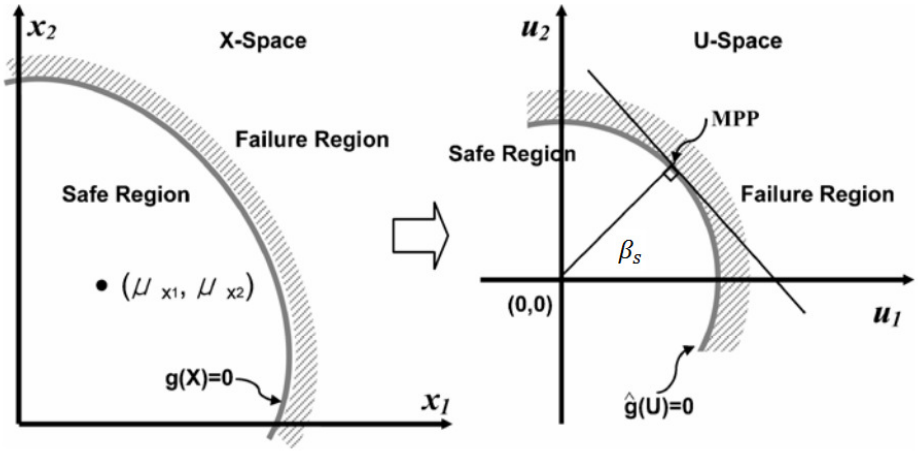


Figure 2.4 : Mapping of failure surface from X-space to U-space [50]

2.4 Reliability Index Approach

In RIA, reliability index β_{s_i} can be obtained using mean value first order second moment (MVFOSM), FORM or SORM. β_{s_i} can be obtained using FORM by formulating an optimization problem with one equality constraint in *U*-space follows as;

$$Minimize \ : ||\mathbf{U}||$$

$$Subject\ to: \ g(\mathbf{U}) = 0 \quad \textbf{(2.37)}$$

where the optimum point on the failure surface is called the most probable failure point (MPP). There are many algorithms available that can solve this problem [52]. In this thesis; the most commonly used recursive algorithms, the Hasofer Lind (HL)

and Hasofer Lind - Rackwitz Fiessler (HL-RF) methods, are introduced to solve the reliability problems.

2.4.1 First Order Reliability Method (FORM)

In this section, we first discuss the MVFOSM method, and then the details of FORM, because the development of FORM can be traced to MVFOSM.

Mean Value First Order Second Moment (MVFOSM) Method

The name "first-order" comes from the first-order expansion of the performance function. In the MVFOSM method, the performance function is represented as the first-order Taylor series expansion at the mean value point. Assuming that the random variables $\boldsymbol{X}$ are statistically independent, the approximate performance function at the mean is written as

$$\tilde{g}(\boldsymbol{X}) \approx g(\mu_{\boldsymbol{X}}) + \boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})^T (\boldsymbol{X} - \mu_{\boldsymbol{X}}) \quad \textbf{(2.38)}$$

where, $\mu_{\boldsymbol{X}} = \{\mu_{x_1}, \mu_{x_2}, \dots, \mu_{x_n}\}^T$ is the mean value vector, and $\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})$ is the gradient vector of g evaluated at $\mu_{\boldsymbol{X}}$, $\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}}) = \left\{\frac{\partial g(\mu_{\boldsymbol{X}})}{\partial x_1}, \frac{\partial g(\mu_{\boldsymbol{X}})}{\partial x_2}, \dots, \frac{\partial g(\mu_{\boldsymbol{X}})}{\partial x_{\mathrm{n}}}\right\}^T$.

The mean value of the approximate performance function $\tilde{g}(\boldsymbol{X})$ is

$$\mu_{\tilde{g}} \approx E[\tilde{g}(\boldsymbol{X})] = g(\mu_{\boldsymbol{X}}) \quad \textbf{(2.39)}$$

The variance of the approximate performance function $\tilde{g}(\boldsymbol{X})$ is

$$Var[\tilde{g}(\boldsymbol{X})] \approx Var[g(\mu_{\boldsymbol{X}})] + Var[\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})^T (\boldsymbol{X} - \mu_{\boldsymbol{X}})] \quad \textbf{(2.40)}$$

In Equation (2.40),

$$Var[g(\mu_{\boldsymbol{X}})] = 0$$

$$Var[\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})] = 0$$

$$Var[\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})^T (\boldsymbol{X} - \mu_{\boldsymbol{X}})] = Var[\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})^T \boldsymbol{X}] - \mu_{\boldsymbol{X}} Var[\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})]$$

$$Var[\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})^T (\boldsymbol{X} - \mu_{\boldsymbol{X}})] = Var[\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})^T \boldsymbol{X}] - 0 = [\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})^T]^2 Var(\boldsymbol{X})$$

$$Var[\tilde{g}(\boldsymbol{X})] = [\boldsymbol{\nabla} g(\mu_{\boldsymbol{X}})^T]^2 Var(\boldsymbol{X}) \quad \textbf{(2.41)}$$

Therefore, the standard deviation of the approximate performance function is

$$\sigma_{\tilde{g}} = \sqrt{Var[\tilde{g}(\boldsymbol{X})]} = \sqrt{[\nabla g(\mu_{\boldsymbol{X}})^T]^2 Var(\boldsymbol{X})} = \left[\sum_{i=1}^{n} \left(\frac{\partial g(\mu_{\boldsymbol{X}})}{\partial x_{\mathrm{i}}}\right)^2 \sigma_{x_i}^2\right]^{\frac{1}{2}} \quad \textbf{(2.42)}$$

The reliability index β_s is computed as

$$\beta_s = \frac{\mu_{\tilde{g}}}{\sigma_{\tilde{g}}} = \frac{g(\mu_{\boldsymbol{X}})}{\left[\sum_{i=1}^{n}\left(\frac{\partial g(\mu_{\boldsymbol{X}})}{\partial x_i}\right)^2 \sigma_{x_i}^2\right]^{\frac{1}{2}}} \quad \textbf{(2.43)}$$

If the performance function is linear Equation (2.43) is same as with Equation (2.22). If the performance function is nonlinear, the approximate performance function $\tilde{g}(\boldsymbol{X})$ is obtained by linearizing the original performance function $g(\mu_{\boldsymbol{X}})$ at the mean value point. This method is called the MVFOSM method, and the β_s given in Equation (2.43) is called a MVFOSM reliability index.

The MVFOSM method changes the original complex probability problem into a simple problem. However, there are two serious drawbacks in the MVFOSM method:

1- Evaluation of reliability by linearizing the limit-state function about the mean values leads to erroneous estimates for performance functions with high nonlinearity, or for large coefficients of variation [50].
2- The MVFOSM method fails to be invariant with different mathematically equivalent formulations of the same problem. This is a problem not only for nonlinear forms of $g(.)$, but also for certain linear forms [50].

Hasofer and Lind (HL) Reliability Index

The reliability index that can be shown from Figure 2.3, is the measure of the distance from the origin to the failure surface. In the one-dimensional case, the standard deviation of the performance function $g(.)$ is used as the scale. To obtain a similar scale in multiple variables, Hasofer and Lind [17] proposed a linear mapping of the basic variables into a set of normalized and independent variables, u_i.

Assess the fundamental case with the independent variables of strength, R, and stress, S, that are both normally distributed.

1- Hasofer and Lind presents the standard normalized random variables,

$$\hat{R} = \frac{R-\mu_R}{\sigma_R}, \hat{S} = \frac{S-\mu_S}{\sigma_S} \quad \textbf{(2.44)}$$

where μ_R and μ_S are the mean values of random variables R and S, respectively, and σ_R and σ_S are the standard deviations of R and S, respectively.

2- Transform the failure surface $g(R,S)=0$ in the original (R,S) coordinate system into the failure surface $\hat{g}(\hat{R},\hat{S})$ in the standard normalized $(\hat{R},\hat{S})$ coordinate system,

$$g\left(R(\hat{R}),S(\hat{S})\right)=\hat{g}(\hat{R},\hat{S})=\hat{R}\sigma_R-\hat{S}\sigma_S+(\mu_R-\mu_S)=0 \quad \textbf{(2.45)}$$

3- In the $(\hat{R},\hat{S})$ coordinate system, the shortest distance from the origin to the failure surface $\hat{g}(\hat{R},\hat{S})=0$ is equal to the reliability index as shown in Figure 2.5.

$$\beta_s=\hat{O}P^*=\frac{\mu_R-\mu_S}{\sqrt{\sigma_R^2+\sigma_S^2}} \quad \textbf{(2.46)}$$

The point $\boldsymbol{P}^*(\hat{\boldsymbol{R}}^*,\hat{\boldsymbol{S}}^*)$ on $\hat{g}(\hat{R},\hat{S})=0$ which corresponds to this shortest distance, is often referred to as the most probable failure point (MPP).

In normally distributed and independent variables of n-dimensional space, the failure surface is a nonlinear function:

$$g(\boldsymbol{X})=g(\{x_1,x_2,\dots,x_n\}^T)=0 \quad \textbf{(2.47)}$$

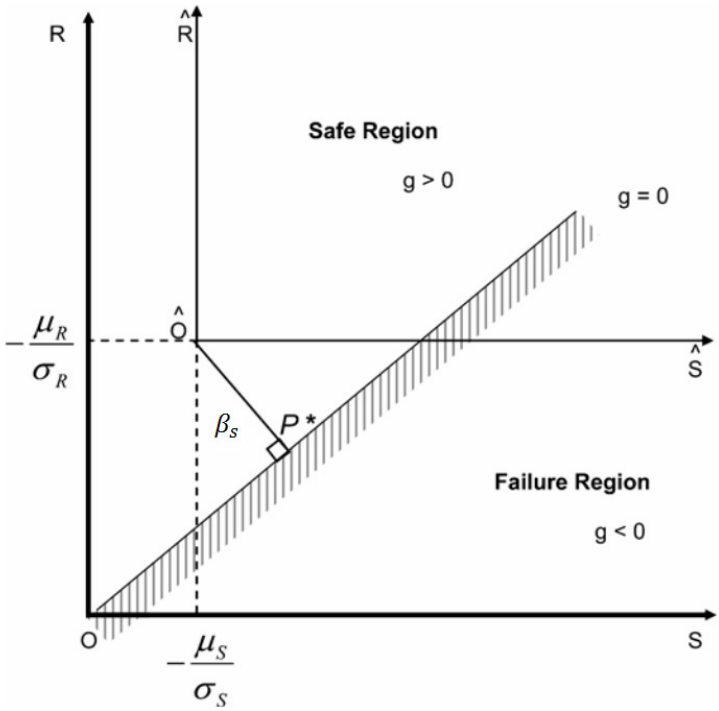


Figure 2.5 : Geometrical illustration of reliability index β_s [50]

The failure surface $g(\boldsymbol{X})=0$ in X-space is mapped into the corresponding failure surface $g(\boldsymbol{U})=0$ in U-space, as shown in Figure 2.4. The reliability index β_s is the shortest distance from the origin to the failure surface $g(\boldsymbol{U})=0$ as

$$\beta_s = \min_{\boldsymbol{U} \in g(\boldsymbol{U}) = 0} (\boldsymbol{U}^T \boldsymbol{U})^{\frac{1}{2}} = \min_{\boldsymbol{U} \in g(\boldsymbol{U}) = 0} \|\boldsymbol{U}\|_2 \quad \textbf{(2.48)}$$

This reliability index β_s is also called the Hasofer and Lind (HL) reliability index.

Hasofer and Lind (HL) Iteration Method

From Equation (2.48), reliability index, β_s, is the solution of a constrained optimization problem in the standard normal space.

$$Minimize \ : \beta(\mathbf{U}) = \|\mathbf{U}\|_2$$

$$Subject\ to: \ g(\mathbf{U}) = 0 \quad \textbf{(2.49)}$$

Suppose that the failure surface with n-dimensional normally distributed and independent random variables $\boldsymbol{X}$ is

$$g(\boldsymbol{X}) = g(\{x_1, x_2, \dots, x_n\}^T) = 0 \quad \textbf{(2.50)}$$

This performance function can be linear or nonlinear. Transforming the performance function given in Equation (2.50) using Equation (2.36),

$$g(\boldsymbol{U}) = g\left(\{\sigma_{X_1} u_1 + \mu_{X_1}, \sigma_{X_2} u_2 + \mu_{X_2}, \dots, \sigma_{X_n} u_n + \mu_{X_n}\}^T\right) = 0 \quad \textbf{(2.51)}$$

The first-order Taylor series expansion of $g(\boldsymbol{U})$ at the MPP $\boldsymbol{P}^*$ is

$$\tilde{g}(\boldsymbol{U}) \approx g(\boldsymbol{u}^*) + \sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial u_i}(u_i - u_i^*) \quad \textbf{(2.52)}$$

From the transformation of Equation (2.36), we have

$$\frac{\partial g(\boldsymbol{u}^*)}{\partial u_i} = \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i} \quad \textbf{(2.53)}$$

The shortest distance from the origin to the approximate failure surface given in Equation (2.52) is

$$\beta_s = \frac{\mu_{\tilde{g}}}{\sigma_{\tilde{g}}} = \frac{g(\boldsymbol{u}^*) - \sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i} u_i^*}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\right)^2 \sigma_{X_i}^2}} \quad \textbf{(2.54)}$$

The derivation of Equation (2.54) is shown in Appendix A.2. The direction cosine of the unit outward normal vector is given as

$$\alpha_i = cos\ \theta_i = -\frac{\frac{\partial g(\boldsymbol{u}^*)}{\partial u_i}}{\|\boldsymbol{\nabla} g(\boldsymbol{u}^*)\|} = -\frac{\frac{\partial g(\boldsymbol{x}^*)}{\partial x_i}\sigma_{X_i}}{\sqrt{\sum_{i=1}^{n}\left(\frac{\partial g(\boldsymbol{x}^*)}{\partial x_i}\right)^2 \sigma_{X_i}^2}} \quad \textbf{(2.55)}$$

where α_i expresses the relative effect of the corresponding random variable on the total variation as shown in Figure 2.6. Thus, it is called the sensitivity factor. The relationship between β_s and α_i is given as

$$\frac{\partial \beta_s}{\partial u_i} = \frac{\partial}{\partial u_i}\sqrt{u_1^2 + u_2^2 +, \dots, + u_n^2} = \frac{u_i}{\beta_s} = \alpha_i\,, \quad (i = 1,2, \dots, n) \tag{2.56}$$

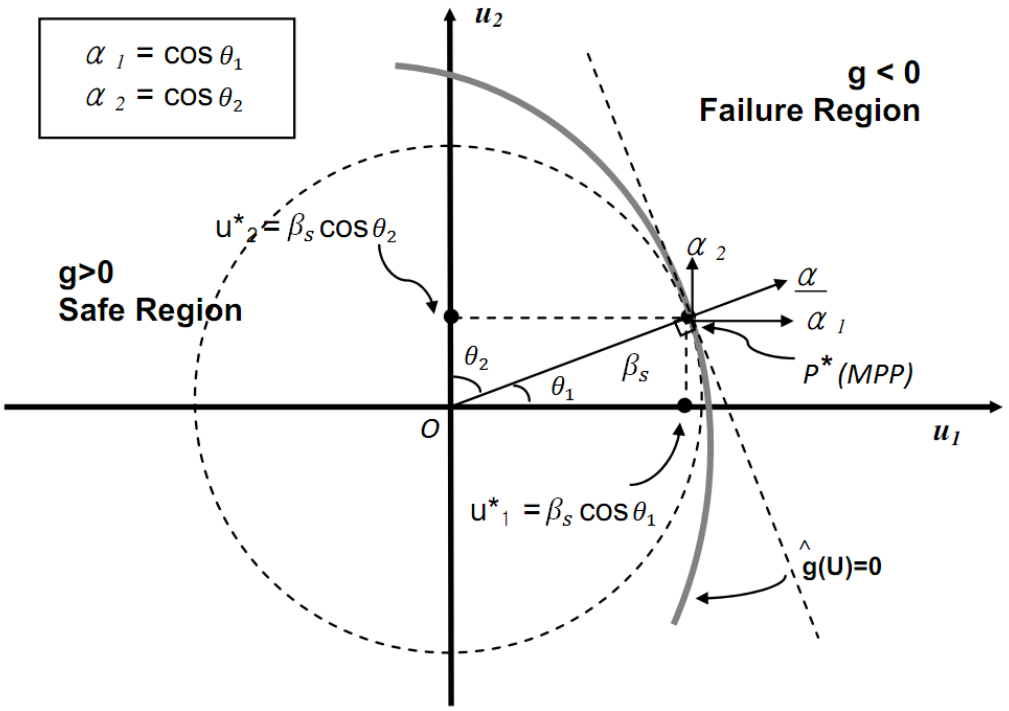


Figure 2.6 : Sensitivity factors [50]

The coordinates of the MPP $\boldsymbol{P}^*$ are computed from Equations (2.54) and (2.55),

$$u_i^* = \frac{x_i^* - \mu_{X_i}}{\sigma_{X_i}} = \beta_s\, \alpha_i \tag{2.57}$$

The coordinates corresponding to MPP $\boldsymbol{P}^*$ in the original space are

$$x_i^* = \mu_{X_i} + \beta_s\, \sigma_{X_i}\, \alpha_i\,, \quad (i = 1,2, \dots, n) \tag{2.58}$$

Since MPP $\boldsymbol{P}^*$ is a point on the limit-state surface,

$$g(\boldsymbol{u}^*) = g(\{x_1^*, x_2^*, \dots, x_n^*\}^T) = 0 \tag{2.59}$$

The main steps of the HL iteration method are:

1- Define the appropriate performance function of Equation (2.50)

2- Assign the mean value point as an initial design point,

$$x_{i,k} = \mu_{X_i}\,, \quad (i = 1,2, \dots n) \tag{2.60}$$

and compute the gradients $\boldsymbol{\nabla} g(\boldsymbol{X}_k)$ of the performance function at this point. $x_{i,k}$ is the i^{th} element in the vector $\boldsymbol{X}_k$ of the k^{th} iteration

3- Compute the initial β_s (MVFOSM reliability index) using Equation (2.43) and its sensitivity factor using Equation (2.55)
4- Compute a new design point $\boldsymbol{X}_k$ and $\boldsymbol{U}_k$ using Equations (2.58) and (2.57), function value, and gradients at this new design point
5- Compute the reliability index β_s (HL reliability index) using Equation (2.54) and its sensitivity factor using Equation (2.55)
6- Iterate steps 4)~6) until the estimate of β_s converges
7- Calculate the coordinates of the design point $\boldsymbol{X}_k$ or MPP, $\boldsymbol{x}^*$.

In this work, the partial derivatives needed for direction cosine and β_s calculations will be computed by finite differencing since commercial codes are used to solve the structural and aerodynamic responses. These commercial codes will be treated as black-box and run again with a small perturbation parameter from batch mode so that the numerical derivative needed in reliability analysis will be automatically computed by forward differencing in HL iterations in reliability analysis.

Hasofer Lind – Rackwitz Fiessler (HL-RF) Method

The HL method was proposed by Hasofer and Lind. The HL method only considers normally distributed random variables, so it cannot be used for non-normal random variables. In non-normal cases, the probability of failure calculation given in Equation (2.28) is inappropriate. Rackwitz and Fiessler extended the HL method to include random variable distribution information, calling their extended method the HL-RF method.

A simple transformation called the equivalent normal distribution (normal tail approximation) is described below. When the random variables are mutually independent, the transformation is given as

$$u_i = \Phi^{-1}\left[F_{X_i}(x_i)\right] \tag{2.61}$$

where $\Phi^{-1}[.]$ is the inverse of the standard normal cumulative distribution function $\Phi[.]$.

Rosenblatt transformation [50] can be used to obtain equivalent normal distribution. Equivalent normal distribution can be obtained by matching the cumulative distribution functions and probability density functions of the original, non-normal

random variable distributions, and the approximate or equivalent normal random variable distributions at the MPP.

Assuming that x_i' is an equivalent normally distributed random variable, the CDF values of x_i and x_i' are equal at the MPP:

$$F_{X_i}(x_i^*) = F_{X_i'}(x_i^*) = \Phi\left(\frac{x_i^* - \mu_{X_i'}}{\sigma_{X_i'}}\right) \quad \textbf{(2.62)}$$

So

$$\mu_{X_i'} = x_i^* - \Phi^{-1}\left[F_{X_i}(x_i^*)\right]\sigma_{X_i'} \quad \textbf{(2.63)}$$

The PDF values of x_i and x_i' at x_i^* are equal:

$$f_{X_i}(x_i^*) = f_{X_i'}(x_i^*) = \frac{1}{\sigma_{X_i'}}\phi\left(\frac{x_i^* - \mu_{X_i'}}{\sigma_{X_i'}}\right) \quad \textbf{(2.64)}$$

From Equations (2.63) and (2.64), the equivalent mean $\mu_{X_i'}$ and standard deviation $\sigma_{X_i'}$ of the approximate normal distribution is derived as below:

$$\sigma_{X_i'} = \frac{\phi\left(\Phi^{-1}\left[F_{X_i}(x_i^*)\right]\right)}{f_{X_i}(x_i^*)} \quad \textbf{(2.65)}$$

$$\mu_{X_i'} = x_i^* - \Phi^{-1}\left[F_{X_i}(x_i^*)\right]\sigma_{X_i'} \quad \textbf{(2.66)}$$

This normal tail approximation is shown in Figure 2.7.

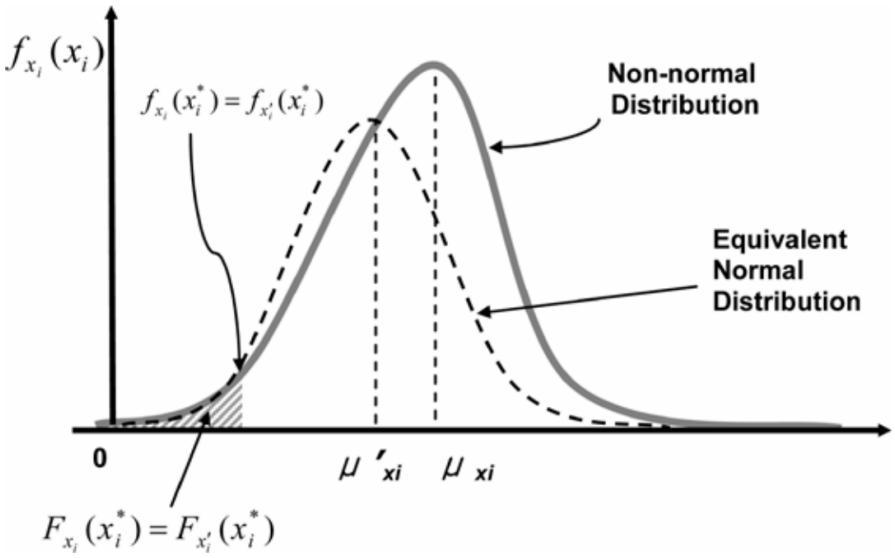


Figure 2.7 : Normal tail approximation [50]

The transformation of the random variables from the X-space to the U-space can be performed using Equations (2.65), (2.66) and (2.67),

$$u_i = \frac{x_i - \mu_{X_i'}}{\sigma_{X_i'}} \tag{2.67}$$

and the performance function $g(\boldsymbol{U})$ in U-space is approximately obtained.

The HL-RF algorithm is similar to the HL iteration method. A flowchart of the algorithm is given in Figure 2.8.

2.4.2 Example Problem for FORM

The performance function is

$$g(x_1, x_2) = x_1^3 + x_2^3 - 18 \tag{2.68}$$

where x_1 and x_2 are the independent random variables with normal distributions (mean $\mu_{x_1} = \mu_{x_2} = 10.0$, standard deviation $\sigma_{x_1} = \sigma_{x_2} = 5.0$). Find the reliability index β_s by using the FORM [50].

Solution of the example is described below step by step.

Step 1: The mean value point $P_1(10.0, 10.0)$ is set as an initial design point in X-space, it corresponds to point $P_2(0.0, 0.0)$ in U-space after transformation as shown in Figure 2.9.

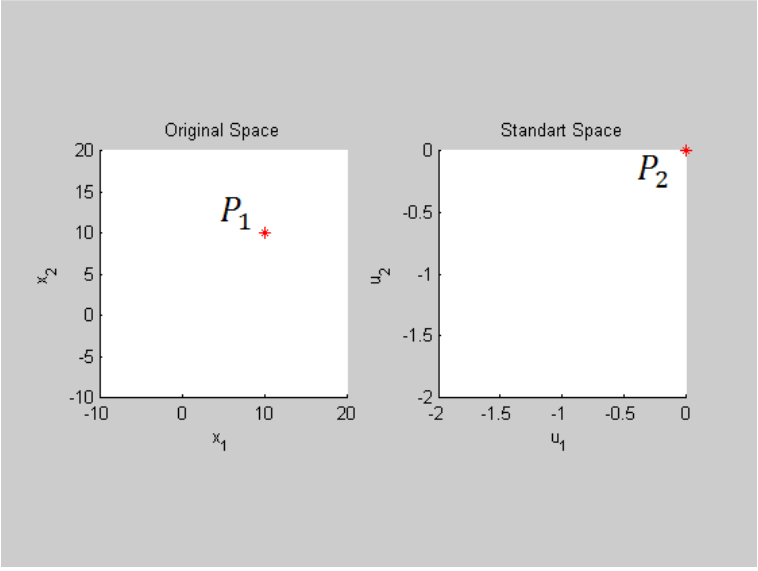


Figure 2.9 : Initial design points in original space (X-space) and standard space (U-space)

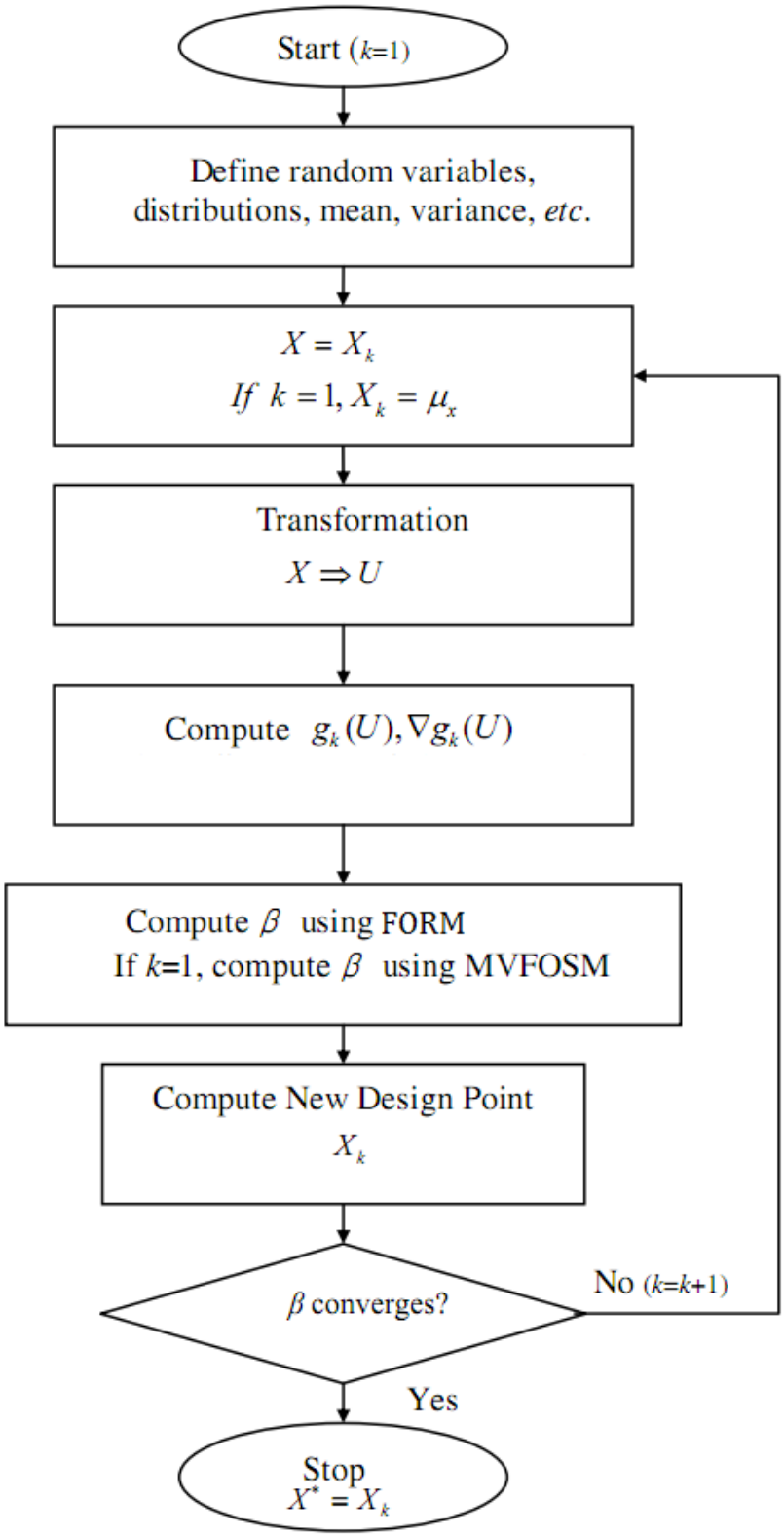


Figure 2.8 : HL-RF method flowchart [50]

Step 2: The first reliability index value $\beta_s = 0.9343$ is computed using MVFOSM, and then the new design point P_3(-0.6607, -0.6607) is obtained as shown in Figure 2.10.

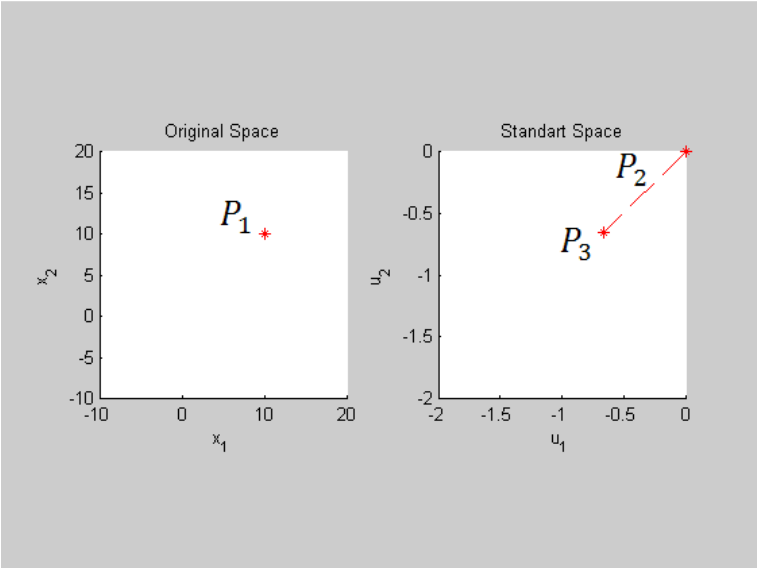


Figure 2.10 : Reliability index β_s and new design point P_3

Step 3: In Figure 2.11, a circle with a radius $\beta_s = 0.9343$ and P_2(0.0, 0.0) as a center point is drawn. The performance function's value at point P_3(-0.6607, -0.6607) is $g(u_1, u_2) = 582.629$. The point P_3(-0.6607, -0.6607) in U-space corresponds to P_4(6.6967, 6.6967) in X-space with the same performance function value $g(x_1, x_2) = 582.629$.

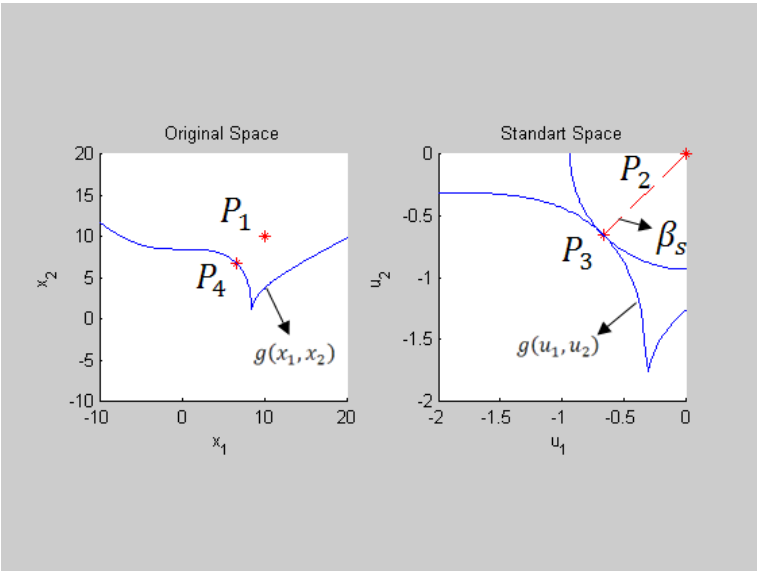


Figure 2.11 : Reliability index β_s and performance function g

Step 4: The second reliability index value $\beta_s = 1.5468$ is computed using FORM, and then the new design point P_5(-1.0937, -1.0937) is obtained as shown in Figure 2.12.

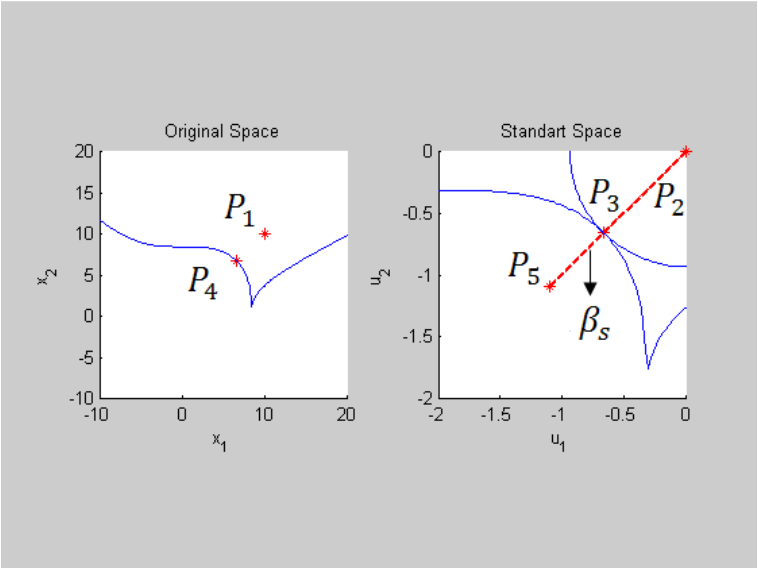


Figure 2.12 : Reliability index β_s and new design point P_5

The same procedures can be repeated until the estimate of β_s converges. The iteration results are summarized in Table 2.1. The reliability index is $\beta_s = 2.2401$. Most probable failure point (MPP) corresponding to this reliability index is $x^* = [\,2.0801, 2.0801\,]$.

Table 2.1 : Iteration results in FORM

Iteration No.	1	2	3	4	5	6	7
$g(x_1, x_2)$	1982.0	582.629	168.08	45.529	10.01	1.1451	0.023
$\frac{\partial g}{\partial x_1}$	300	134.537	61.598	30.0897	17.43	13.5252	12.9917
$\frac{\partial g}{\partial x_2}$	300	134.537	61.598	30.0897	17.43	13.5252	12.9917
β_s	0.9343	1.5468	1.9327	2.1467	2.2279	2.2398	2.2401
α_1	-0.7071	-0.7071	-0.7071	-0.7071	-0.7071	-0.7071	-0.7071
α_2	-0.7071	-0.7071	-0.7071	-0.7071	-0.7071	-0.7071	-0.7071
x_1	6.6967	4.5313	3.1670	2.4104	2.1233	2.0810	2.0801
x_2	6.6967	4.5313	3.1670	2.4104	2.1233	2.0810	2.0801
u_1	-0.6607	-1.0937	-1.3666	-1.5179	-1.5753	-1.5838	-1.5840
u_2	-0.6607	-1.0937	-1.3666	-1.5179	-1.5753	-1.5838	-1.5840
Convergence (ε)	-	0.6556	0.2495	0.1107	0.036	0.005	0.00001

2.5 Performance Measure Approach

In PMA, reliability analysis can be formulated as the inverse of reliability analysis in RIA. The probabilistic performance measure g_{p_i} can be obtained using MVFOSM method, FORM or SORM. Here, g_{p_i} is obtained using the FORM in *U*-space defined as

$$Minimize \ : g(\mathbf{U})$$

$$Subject\ to : ||\mathbf{U}|| = \beta_t \tag{2.69}$$

where the optimum point on a target reliability surface is identified as the MPP. General optimization algorithms can be employed to solve the optimization problem in Equation (2.69). However, the Advanced Mean Value (AMV) method is well suited for PMA due to its simplicity and efficiency [39].

2.5.1 Advanced Mean Value (AMV) Method

Formulation of the AMV method begins with the MVFOSM method, defined as

$$\boldsymbol{u}^*_{MVFOSM} = \beta_t\, \boldsymbol{n}(0) \quad \text{where} \quad \boldsymbol{n}(0) = -\frac{\nabla_{\boldsymbol{X}} g(\mu_{\boldsymbol{X}})}{\|\nabla_{\boldsymbol{X}} g(\mu_{\boldsymbol{X}})\|} = -\frac{\nabla_{\boldsymbol{U}} g(0)}{\|\nabla_{\boldsymbol{U}} g(0)\|} \tag{2.70}$$

To minimize the performance function $g(\mathbf{U})$ in Equation (2.70), the normalized steepest descent direction $\boldsymbol{n}(0)$ is defined at the mean value. The AMV method iteratively updates the direction vector of the steepest descent method at the probable point $\boldsymbol{u}^{(k)}_{AMV}$ initially obtained using the MVFOSM method. Thus, the AMV method can be formulated as

$$\boldsymbol{u}^{(1)}_{AMV} = \boldsymbol{u}^*_{MVFOSM}\ , \qquad \boldsymbol{u}^{(k+1)}_{AMV} = \beta_t\, \boldsymbol{n}(\boldsymbol{u}^{(k)}_{AMV}) \tag{2.71}$$

where

$$\boldsymbol{n}\left(\boldsymbol{u}^{(k)}_{AMV}\right) = -\frac{\nabla_{\boldsymbol{U}} g(\boldsymbol{u}^{(k)}_{AMV})}{\left\|\nabla_{\boldsymbol{U}} g(\boldsymbol{u}^{(k)}_{AMV})\right\|} \tag{2.72}$$

2.5.2 Example Problem for AMV Method

The performance function is

$$g(x_1, x_2) = -\exp(x_1 - 7) - x_2 + 10 \quad \textbf{(2.73)}$$

where x_1 and x_2 are the independent random variables with normal distributions (mean $\mu_{x_1} = \mu_{x_2} = 6.0$, standard deviation $\sigma_{x_1} = \sigma_{x_2} = 0.8$). The target reliability index is set to $\beta_t = 3.0$. Find the performance measure g by using the AMV method [39].

Solution of the example is described below step by step.

Step 1: The mean value point $P_1(6.0, 6.0)$ is set as an initial design point in X-space, it corresponds to point $P_2(0.0, 0.0)$ in U-space after transformation as shown in Figure 2.13.

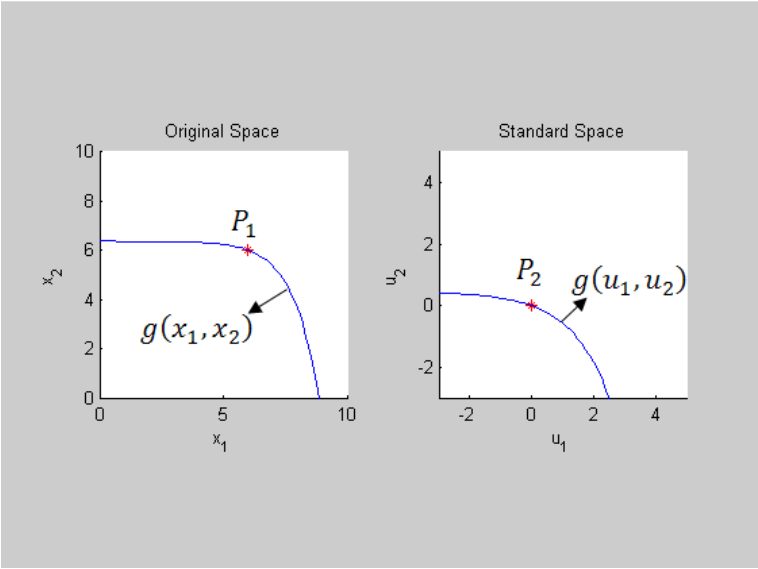


Figure 2.13 : Initial design points in original space (X-space) and standard space (U-space)

Step 2: In Figure 2.14, a circle with a radius $\beta_t = 3.0$ and $P_2(0.0, 0.0)$ as a center point is drawn. The new design point is obtained as $P_3(1.0358, 2.8155)$. The performance measure value at point P_3 is $g(u_1, u_2) = 0.9051$. The point P_3 in U-space corresponds to $P_4(6.8286, 8.2524)$ in X-space with the same performance measure value $g(x_1, x_2) = 0.9051$.

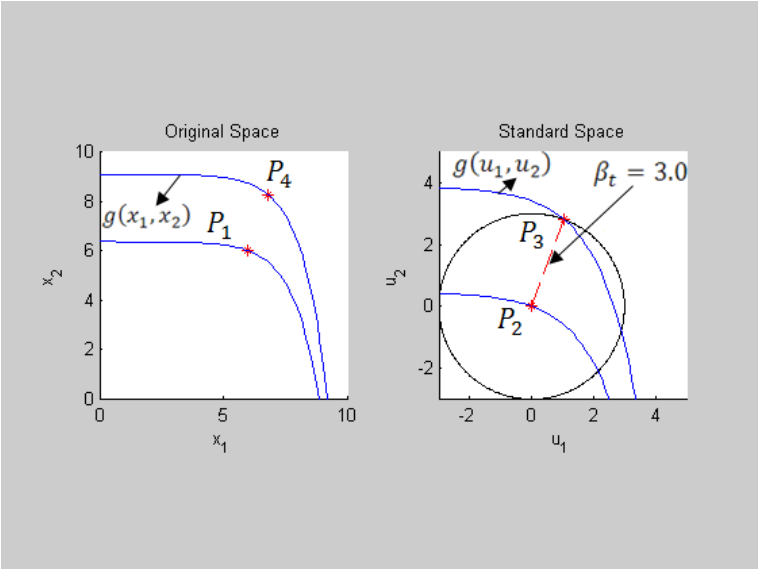


Figure 2.14 : Performance function g and new design point P_3

Step 3: In the next iteration, the design point is obtained as P_5(1.9329, 2.2943) and the performance measure value at the point P_5 is $g(u_1, u_2) = 0.4376$ as shown in Figure 2.15.

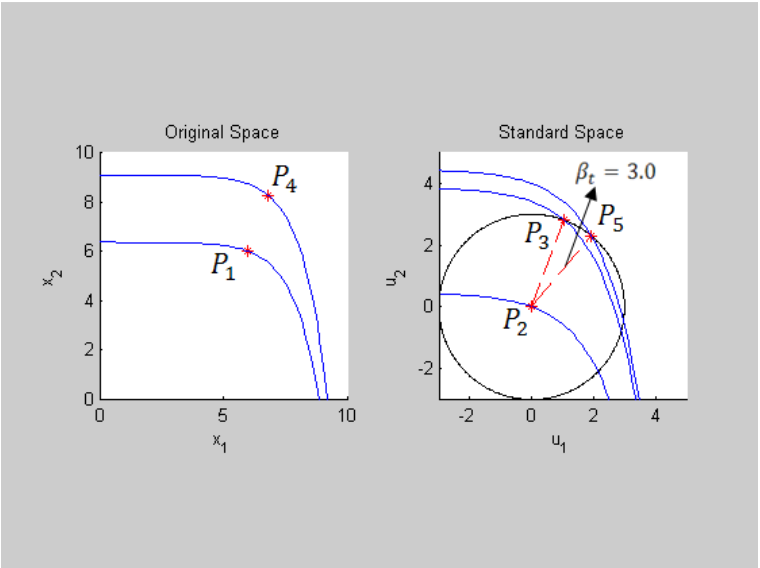


Figure 2.15 : Design point P_5 and target reliability index β_t

The same procedures can be repeated until the estimate of performance measure g converges. The iteration results are summarized in Table 2.2. The performance measure is $g = -0.3579$. Most probable failure point (MPP) corresponding to this performance measure is $x^* = [\,8.3173, 6.6247\,]$.

Table 2.2 : Iteration results in AMV method

Iteration No.	1	2	3	4	5	6
β_t	3.0	3.0	3.0	3.0	3.0	3.0
$\frac{\partial g}{\partial x_1}$	-0.2943	-0.6740	-1.3816	-2.3485	-2.8538	-2.9677
$\frac{\partial g}{\partial x_2}$	-0.8000	-0.8000	-0.8000	-0.8000	-0.8000	-0.8000
$g(x_1, x_2)$	0.9051	0.4376	-0.1383	-0.3412	-0.3574	-0.3579
n_1	0.3453	0.6443	0.8654	0.9466	0.9629	0.9655
n_2	0.9385	0.7648	0.5011	0.3224	0.2699	0.2603
x_1	6.8286	7.5463	8.0769	8.2718	8.3109	8.3173
x_2	8.2524	7.8354	7.2027	6.7739	6.6478	6.6247
u_1	1.0358	1.9329	2.5962	2.8398	2.8886	2.8966
u_2	2.8155	2.2943	1.5033	0.9673	0.8098	0.7808
Convergence (ε)	-	0.4674	0.5759	0.2029	0.0162	0.0005

3. RELIABILITY BASED STRUCTURAL OPTIMIZATION OF A CANTILEVER BEAM VERIFICATION

This chapter presents the implementation of the reliability analysis methods using Matlab and a benchmark problem from literature is used to validate the implementation.

3.1 Design of a Cantilever Beam

This test problem is adapted from the reliability-based design optimization literature [53]. Figure 3.1 shows the cantilever beam with a rectangular cross-section subjected to a vertical load, P_Y, and a horizontal load, P_X, at the tip. It is assumed that the beam has a fixed length of $L = 100.0$ *in.* and that the cross sectional dimensions of the beam remain constant along the length of the beam. The design objective is to prevent yielding due to normal stress while minimizing the weight of the beam or, equivalently, the cross-sectional area *w*t*.

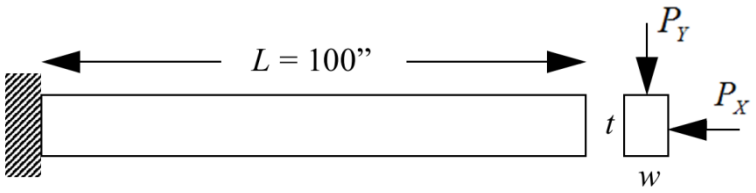


Figure 3.1 : Design optimization of a cantilever beam

We consider yielding at the root of the fixed end of the beam, where the maximum normal stress (S) is calculated analytically as follows

$$S=\left(\frac{600}{wt^2}P_Y+\frac{600}{w^2t}P_X\right) \tag{3.1}$$

The limit state function is;

$$g=R-S=R-\left(\frac{600}{wt^2}P_Y+\frac{600}{w^2t}P_X\right) \tag{3.2}$$

where R is the random yield strength, P_X and P_Y are mutually independent random loads. Assume that there are uncertainties in P_X, P_Y, and R, and they follow normal distributions with the following parameters as shown in Table 3.1.

Table 3.1 : Statistics of loads and material properties

Random Variable	R (psi)	P_X (lbf)	P_Y (lbf)
Mean	40000	500	1000
Standard Deviation	2000	100	100

It is assumed that the manufacturing tolerances on the cross-sectional width w and thickness t are relatively small and, therefore, the dimensions will be treated as deterministic design variables.

3.2 Deterministic Design

In deterministic design, the design with minimum weight can be obtained by solving the following optimization problem using a safety factor (SF). We assume that the random variables R, P_X and P_Y are fixed at their means.

Minimize : $Area = w * t$

Subject to : $g = R - SF * S \geq 0$ **(3.3)**

3.2.1 Deterministic Analytical Solution

Equation (3.3) is a typical nonlinear programming problem, and is here solved using by a built-in Matlab function called fmincon (Appendix A.3). A safety factor of 1.5 is used for this analytical solution. The maximum normal stress is evaluated using Equation (3.1).

Solution of the reference study took about 2 seconds using Matlab's fmincon function on a workstation with Intel(R) Core(TM) 2 Quad CPU Q8300@2.50 GHz processor, with 2.00 GB of RAM on Microsoft Windows XP operating system. As a result, the optimum design shown in Table 3.2 corresponding to the minimum area is $(w, t) = (2.2407, 4.4814)$.

Table 3.2 : Deterministic analytical design result

w (inch)	t (inch)	g (psi)	$Area\ (inch^2)$
2.2407	4.4814	0.1271	10.0415

3.2.2 Deterministic Computational Solution

For a problem which the analytical formulation of stress function is not available, the stress value can be calculated from the finite element analysis software, and here Abaqus is used to obtain the maximum normal stress value (S). This exercise is a building block for the more complicated applications that will be solved in this thesis.

Equation (3.3) is solved using Modefrontier as an optimizer driver. The optimization workflow for deterministic design can be shown in Figure 3.2.

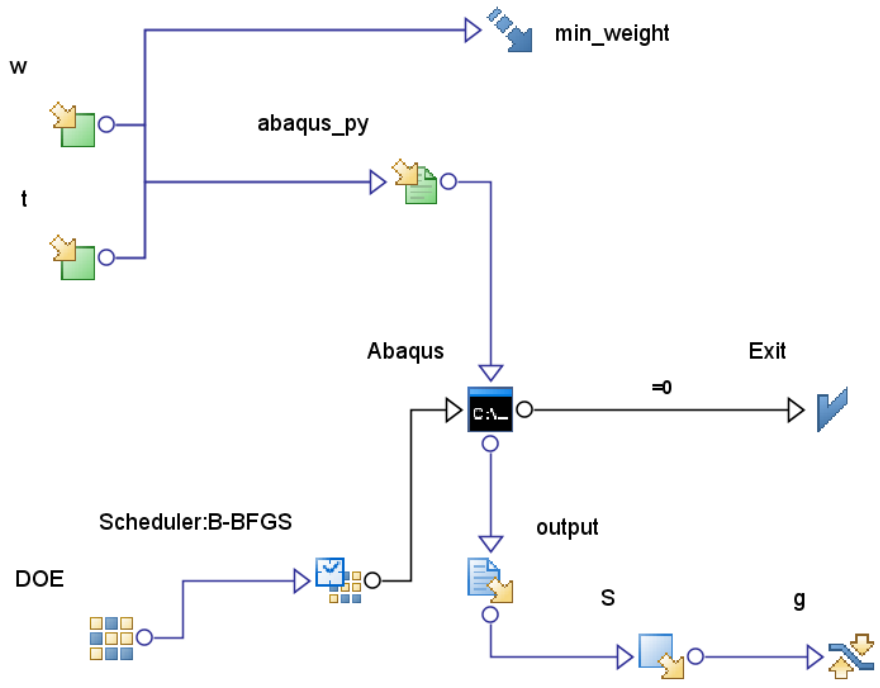


Figure 3.2 : Optimization workflow for deterministic design

Solution of the present study took about 5 hours 37 minutes using Modefrontier on a workstation with Intel(R) Core(TM) 2 Quad CPU Q8300@2.50 GHz processor, with 2.00 GB of RAM on Microsoft Windows XP operating system. Design of experiment (DOE) techniques are used to limit the number of runs. Initial number of designs are provided for the learning process of the algorithm so optimum designs can be obtained fastly. In this study, 16 DOE is used. “Full Factorial” method which

works best with less than 8 variables is employed to distribute DOE points. BFGS (Broyden-Fletcher-Goldfarb-Shanno) algorithm which is a method for solving nonlinear optimization problems is used for attaining optimum result. Finally a total number of 1157 designs are generated for the optimization problem. As a result 209 unfeasible designs and 948 feasible designs are found. The values of selected designs are given in Table 3.3. The first design which has the minimum area is the optimum solution.

Table 3.3 : Selected designs of deterministic computational solution

Design	w (inch)	t (inch)	g (psi)	$Area\ (inch^2)$
1	2.2135	4.5366	0.55	10.041
2	2.2824	4.4000	1.6	10.042
3	2.1962	4.5729	2.95	10.043

3.2.3 Comparison of Analytical and Computational Solutions

Both the analytical and computational results that are obtained using Matlab's fmincon function and Modefrontier are similar, and the results are shown in Table 3.4.

Table 3.4 : Comparison of deterministic design results

	w (inch)	t (inch)	g (psi)	$Area\ (inch^2)$
Present Study ~ Modefrontier	2.2135	4.5366	0.55	10.041
Reference Study ~ Literature (fmincon) [53]	2.2407	4.4814	0.1271	10.0415

3.3 Reliability Based Design Optimization

3.3.1 Reliability Index Approach (RIA)

Depending on the goal of the optimization problem, different formulations can be used. For example, if the goal is to achieve maximum reliability as long as the weight is within some bounds, the optimization problem can be formulated as

Maximize : β_s

Subject to : $Area = w * t \leq Area\ upper\ bound$ **(3.4)**

If the goal is to achieve minimum weight as long as the reliability is within some bounds, the optimization problem can be formulated as

Minimize : $Area = w * t$

Subject to : $\beta_s \geq \beta_t$ **(3.5)**

The selection of a target reliability index, β_t, is problem dependent and often controversial. A commonly used value is 3.000, corresponding to, for a normally distributed performance function (g), a reliability of 0.99865 or a probability of failure of 0.00135.

Equation (3.5) is used in this study to optimize the beam design. The β_s value is computed using first order reliability method (FORM) for a given design with both analytical and computational formulation of stress function.

3.3.1.1 Reliability Index Approach with Analytical Stress Solution

From literature, we have observed that the stress constraint is usually dominant with respect to the displacement constraint. For approval of this condition, in RBDO, in addition to stress constraint a displacement constraint is also considered in the optimization problem. Analytical formulation of the problem is

Minimize : $Area = w * t$

Subject to : $\beta_s \geq \beta_t = 3.000$

$\beta_d \geq \beta_t = 3.000$

where $g_{stress} = R - S = R - \left(\frac{600}{wt^2} P_Y + \frac{600}{w^2 t} P_X \right)$

$$g_{displacement} = D_0 - D = D_0 - \frac{4L^3}{Ewt} \sqrt{\left(\frac{P_Y}{t^2} \right)^2 + \left(\frac{P_X}{w^2} \right)^2} \quad \textbf{(3.6)}$$

In the above, $D_0 = 2.2535 in.$ is the displacement tolerance at the free end of the beam. E is the random Young's modulus and has a normal distribution of N(29E6, 1.45E6) psi. Equation (3.6) is solved by using a built-in Matlab function, fmincon using FORM (Appendix A.4). Fmincon seeks the optimum solution in the outer loop, FORM is used for reliability analysis in the inner loop.

Solution of the reference study took about 2 seconds using Matlab's fmincon function on a workstation with Intel(R) Core(TM) 2 Quad CPU Q8300@2.50 GHz

processor, with 2.00 GB of RAM on Microsoft Windows XP operating system. As a result, the optimum designs for different constraints are shown in Table 3.5.

Table 3.5 : Reliability index approach with analytical stress solution

Constraints	w (inch)	t (inch)	β_s	β_d	$Area\ (inch^2)$
Stress	2.4460	3.8922	3.000	-	9.5202
Displacement	2.7015	3.4078	-	3.000	9.2063
Stress and Disp.	2.4484	3.8884	3.000	3.000	9.5203

From Table 3.5, the results match well as in the reference study [53] and it is validated that the stress constraint is dominant. In this problem, only stress constraint is considered for the optimization problems for computational simplicity. For the stress constraint, optimum design FORM results are shown in Table 3.6. The reliability index β_s is 3.000. The limit-state function (g) value at most probable failure point (MPP), $x^* = [36705, 712, 1133]$, is zero, this reliability index can be considered as the shortest distance from the origin to the limit-state surface.

Table 3.6 : Iteration results in FORM for the optimum design in analytical RIA

Iteration No.	1	2
$g(R, P_X, P_Y, w, t)$	10925	0
$\frac{\partial g}{\partial R}$	1	1
$\frac{\partial g}{\partial P_X}$	-25.7658	-25.7658
$\frac{\partial g}{\partial P_Y}$	-16.1921	-16.1921
β_s	3.000	3.000
α_1	-0.5492	-0.5492
α_2	0.7076	0.7076
α_3	0.4447	0.4447
R (X-space)	36705	36705
P_X (X-space)	712	712
P_Y (X-space)	1133	1133
R (U-space)	-1.6477	-1.6477
P_X (U-space)	2.1228	2.1228
P_Y (U-space)	1.3340	1.3340
Convergence (ε)	-	2.9605E-16

3.3.1.2 Reliability Index Approach with Computational Stress Solution

Numerical formulation of the problem is

Minimize : $Area = w * t$

Subject to : $\beta_s \geq \beta_t = 3.000$

where $g = R - S$ **(3.7)**

Equation (3.7) is solved using Modefrontier as an optimizer driver. An in-house code developed in Matlab is used for the reliability analysis. Briefly, Modefrontier seeks the optimum solution in the outer loop, FORM code written in Matlab is used for reliability analysis in the inner loop. The optimization workflow for reliability based design can be shown in Figure 3.3. In Appendix A.5, FORM code with the script files written in Matlab is shown. Abaqus calculates the maximum normal stress value (S).

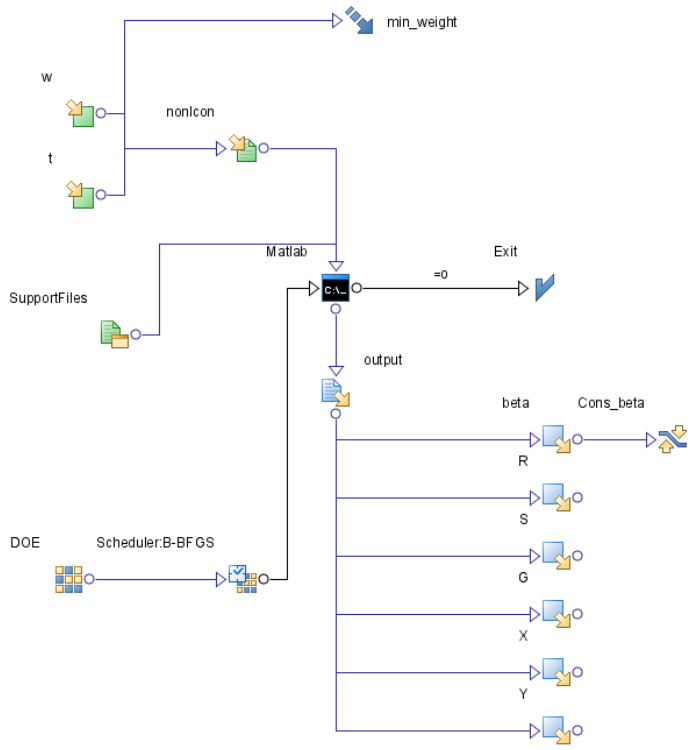


Figure 3.3 : Optimization workflow for reliability index approach (RIA)

Solution of the present study took about 27 hours 13 minutes using Modefrontier on a workstation with Intel(R) Core(TM) 2 Quad CPU Q8300@2.50 GHz processor, with 2.00 GB of RAM on Microsoft Windows XP operating system. In this study, 16 design of experiments (DOE) is used. “Full Factorial” is employed to distribute DOE

points. BFGS algorithm is used to reach the optimum result. Finally a total number of 350 designs are generated during the optimization problem. As a result 67 unfeasible designs and 283 feasible designs are found. The values of selected designs are given in Table 3.7. The first design which has the minimum area is the optimum solution.

Table 3.7 : Selected designs of computational reliability index approach solution

Design	w (inch)	t (inch)	β_s	$Area\ (inch^2)$
1	2.4000	3.9673	3.000	9.5215
2	2.4000	3.9681	3.003	9.5236
3	2.5064	3.8000	3.004	9.5243

For the optimum design FORM results can be shown from Table 3.8. The reliability index β_s is 3.000. The limit-state function (g) value (0.1033) at MPP, $x^* = [36724, 715, 1130]$, is close to zero compared to the starting value, this reliability index can be considered as the shortest distance from the origin to the limit-state surface.

Table 3.8 : Iteration results in FORM for the optimum design in computational RIA

Iteration No.	1	2
$g(R, P_X, P_Y, w, t)$	10990	0.1471
$\frac{\partial g}{\partial R}$	1	1
$\frac{\partial g}{\partial P_X}$	-26.2550	-26.2550
$\frac{\partial g}{\partial P_Y}$	-15.8850	-15.8850
β_s	3.000	3.000
α_1	-0.5460	-0.5460
α_2	0.7168	0.7168
α_3	0.4337	0.4337
R (X-space)	36724	36724
P_X (X-space)	715	715
P_Y (X-space)	1130	1130
R (U-space)	-1.6382	-1.6382
P_X (U-space)	2.1506	2.1506
P_Y (U-space)	1.3012	1.3012
Convergence (ε)	-	1.4E-5

3.3.1.3 Comparison of Analytical and Computational Solutions in RIA

Both the analytical and computational results that are obtained using Matlab's fmincon function and Modefrontier agree well, and the results are shown in Table 3.9.

Table 3.9 : Comparison of analytical and computational solutions in RIA

	w (inch)	t (inch)	β_s	$Area\ (inch^2)$
Present Study ~ Modefrontier	2.4000	3.9673	3.000	9.5215
Reference Study ~ Literature (fmincon) [53]	2.4460	3.8922	3.000	9.5202

Comparison of FORM results for optimum designs in the analytical and computational solutions can be shown from Table 3.10.

Table 3.10 : Comparison of FORM results in analytical and computational solutions for RIA

	Analytical	Computational
$g(R, P_X, P_Y, w, t)$	0	0.1471
$\frac{\partial g}{\partial R}$	1	1
$\frac{\partial g}{\partial P_X}$	-25.7658	-26.2550
$\frac{\partial g}{\partial P_Y}$	-16.1921	-15.8850
β_s	3.000	3.000
α_1	-0.5492	-0.5460
α_2	0.7076	0.7168
α_3	0.4447	0.4337
R (X-space)	36705	36724
P_X (X-space)	712	715
P_Y (X-space)	1133	1130
R (U-space)	-1.6477	-1.6382
P_X (U-space)	2.1228	2.1506
P_Y (U-space)	1.3340	1.3012
Convergence (ε)	2.9605E-16	1.4E-5

3.3.2 Performance Measure Approach (PMA)

3.3.2.1 Performance Measure Approach with Analytical Stress Solution

Analytical formulation of the problem is

Minimize : $Area = w*t$

Subject to : $g = R - S = R - \left(\frac{600}{wt^2} P_Y + \frac{600}{w^2 t} P_X \right) \geq 0$

where $\beta_t = 3.000$ **(3.8)**

Equation (3.8) is solved by using a built-in Matlab function, fmincon using advanced mean value (AMV) method (Appendix A.6). In Equation (3.8), g value is computed using AMV method. Fmincon seeks the optimum solution in the outer loop, AMV method is used for inverse reliability analysis in the inner loop.

Solution of the reference study took about 2 seconds using Matlab's fmincon function on a workstation with Intel(R) Core(TM) 2 Quad CPU Q8300@2.50 GHz processor, with 2.00 GB of RAM on Microsoft Windows XP operating system. As a result, the optimum design shown in Table 3.11 corresponding to the minimum area is (w, t) = (2.4460, 3.8922).

Table 3.11 : Performance measure approach with analytical stress solution

w (inch)	t (inch)	g	$Area$ ($inch^2$)
2.4460	3.8922	$-8.3*10^{-7}$	9.5202

For the optimum design AMV method results can be shown in Table 3.12. The limit-state function (g) value is $-8.3*10^{-7}$ corresponding to a target reliability index $\beta_t = 3.000$. MPP is $x^* = [3670.5, 712.1, 1133]$.

3.3.2.2 Performance Measure Approach with Computational Stress Solution

Numerical formulation of the problem is

Minimize : $Area = w*t$

Subject to : $g = R - S$

where $\beta_t = 3.000$ **(3.9)**

Equation (3.9) is solved using Modefrontier as an optimizer driver. Matlab is used for the inverse reliability analysis. Briefly, Modefrontier seeks the optimum solution in the outer loop, AMV method code written in Matlab is used for inverse reliability analysis in the inner loop. The optimization workflow for reliability based design can be shown in Figure 3.4. In Appendix A.7, AMV method code with the script files

Table 3.12 : Iteration results in AMV method for the optimum design in analytical PMA

Iteration No.	1	2
β_t	3.000	3.000
$\frac{\partial g}{\partial R}$	1	1
$\frac{\partial g}{\partial P_X}$	-25.7659	-25.7659
$\frac{\partial g}{\partial P_Y}$	-16.1924	-16.1924
$g(R, P_X, P_Y, w, t)$	$-8.3*10^{-7}$	$-8.3*10^{-7}$
n_1	-0.5492	-0.5492
n_2	0.7076	0.7076
n_3	0.4447	0.4447
R (X-space)	36705	36705
P_X (X-space)	712	712
P_Y (X-space)	1133	1133
R (U-space)	-1.6477	-1.6477
P_X (U-space)	2.1227	2.1227
P_Y (U-space)	1.3340	1.3340
Convergence (ε)	-	7.2760E-12

written in Matlab is shown. Abaqus is used to compute the maximum normal stress value (S).

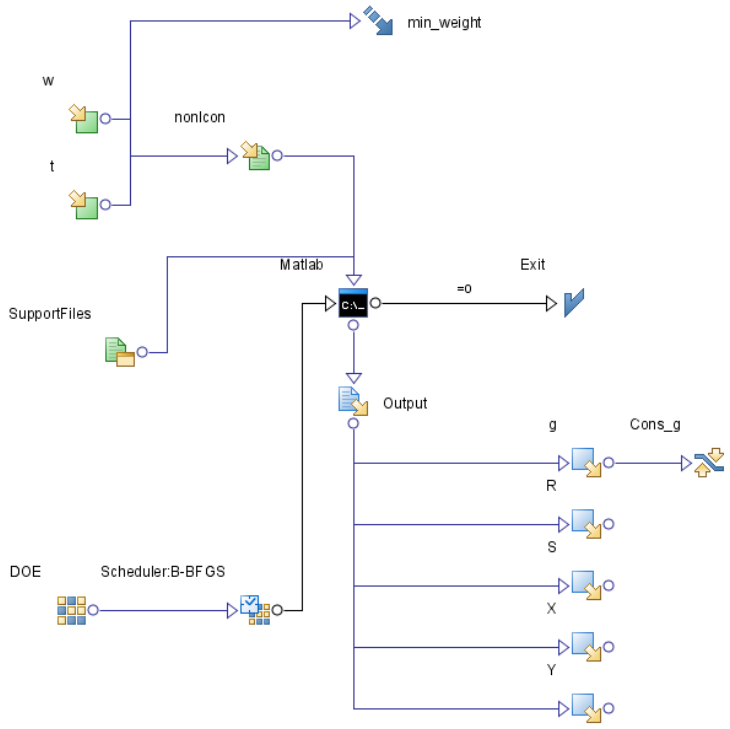


Figure 3.4 : Optimization workflow for performance measure approach (PMA)

Solution of the present study took about 29 hours 10 minutes using Modefrontier on a workstation with Intel(R) Core(TM) 2 Quad CPU Q8300@2.50 GHz processor, with 2.00 GB of RAM on Microsoft Windows XP operating system. In this study, 16 design of experiments (DOE) is used. "Full Factorial" is employed to distribute DOE points. BFGS algorithm is used to get optimum result. Finally a total number of 350 designs are generated for the optimization problem. As a result 67 unfeasible designs and 283 feasible designs are found. The values of selected designs are given in Table 3.13. The first design which has the minimum area is the optimum solution.

Table 3.13 : Selected designs of computational performance measure approach solution

Design	w (inch)	t (inch)	g (psi)	$Area\ (inch^2)$
1	2.4000	3.9673	1.38	9.5215
2	2.4000	3.9681	12.36	9.5236
3	2.5064	3.8000	12.77	9.5243

For the optimum design AMV method results can be shown in Table 3.14. The limit-state function (g) value is 1.3824 corresponding to a target reliability index $\beta_t = 3.000$. MPP is $x^* = [36724, 715, 1130]$.

Table 3.14 : Iteration results in AMV method for optimum design in computational PMA

Iteration No.	1	2
β_t	3.000	3.000
$\frac{\partial g}{\partial R}$	1	1
$\frac{\partial g}{\partial P_X}$	-26.2550	-26.2550
$\frac{\partial g}{\partial P_Y}$	-15.8850	-15.8800
$g(R, P_X, P_Y, w, t)$	1.3763	1.3824
n_1	-0.5460	-0.5461
n_2	0.7168	0.7168
n_3	0.4337	0.4336
R (X-space)	36724	36724
P_X (X-space)	715	715
P_Y (X-space)	1130	1130
R (U-space)	-1.6381	-1.6382
P_X (U-space)	2.1504	2.1505
P_Y (U-space)	1.3010	1.3007
Convergence (ε)	-	0.0061

3.3.2.3 Comparison of Analytical and Computational Solutions

Both the analytical and computational results that are obtained using Matlab's fmincon function and Modefrontier agree well, and the results are shown in Table 3.15.

Table 3.15 : Comparison of analytical and computational solutions in PMA

	w (inch)	t (inch)	g (psi)	$Area$ ($inch^2$)
Present Study ~ Modefrontier	2.4000	3.9673	1.38	9.5215
Reference Study ~ Literature (fmincon) [53]	2.4460	3.8922	$-8.3*10^{-7}$	9.5202

Comparison of optimum results solved by AMV method both with analytical and computational stress solutions are shown in Table 3.16.

Table 3.16 : Comparison of AMV method results in analytical and computational solutions for PMA

	Analytical	Computational
β_t	3.000	3.000
$\frac{\partial g}{\partial R}$	1	1
$\frac{\partial g}{\partial P_X}$	-25.7659	-26.2550
$\frac{\partial g}{\partial P_Y}$	-16.1924	-15.8800
$g(R, P_X, P_Y, w, t)$	$-8.3*10^{-7}$	1.3824
n_1	-0.5492	-0.5461
n_2	0.7076	0.7168
n_3	0.4447	0.4336
R (X-space)	36705	36724
P_X (X-space)	712	715
P_Y (X-space)	1133	1130
R (U-space)	-1.6477	-1.6382
P_X (U-space)	2.1227	2.1505
P_Y (U-space)	1.3340	1.3007
Convergence (ε)	7.2760E-12	0.0061

3.4 Comparison of Deterministic Optimization and Reliability Based Design Optimization

Deterministic design of the cantilever beam results with a bulk beam design both as shown in Table 3.17 and in Table 3.18. RIA and PMA give the same optimum result for this cantilever beam problem in both analytical and computational solutions.

Table 3.17 : Comparison of deterministic and reliability based optimization with analytical stress solutions

	w (inch)	t (inch)	$Area\ (inch^2)$	Safety Factor	Wall Clock Time
Det.	2.2407	4.4814	10.0415	1.5	2 Seconds
RIA	2.4460	3.8922	9.5202	-	2 Seconds
PMA	2.4460	3.8922	9.5202	-	2 Seconds

Comparison of deterministic and reliability based design optimization with computational stress solutions is shown in Table 3.18.

Table 3.18 : Comparison of deterministic and reliability based design optimization with computational stress solutions

	w (inch)	t (inch)	$Area\ (inch^2)$	Safety Factor	Wall Clock Time
Det.	2.2135	4.5366	10.041	1.5	5 hours 37 min.
RIA	2.4000	3.9673	9.5215	-	27 hours 13 min.
PMA	2.4000	3.9673	9.5215	-	29 hours 10 min.

From now on, we deal with structural solutions in the wing problems because analytical formulations of stress and displacement values for complex geometries will not be available. For the beam problem, for each reliability analysis, computational RIA solution calls Abaqus 11 times to obtain the normal stress value while computational PMA solution calls Abaqus 12 times. Because of this state, for one reliability analysis RIA computes the safety index β_s approximately in 4 minutes, PMA computes the probabilistic performance measure g_{p_i} approximately in 5 minutes. After examining Table 3.18, RIA solution is preferred for reliability analysis in the computational wing problems because RIA solution needs less wall clock time to obtain the reliability index value when compared to PMA solution.

4. RELIABILITY BASED DESIGN OPTIMIZATION OF AEROSPACE STRUCTURES

4.1 Generic Aircraft Wing Introduction

Previously, Nikbay et al. [48] presented a practical structural optimization problem on a generic three dimensional wing geometry by employing high fidelity softwares such as Catia, Abaqus and Modefrontier. In this thesis, the previous work will be extended by incorporating an in-house reliability analysis code written in Matlab into that high-fidelity structural optimization framework.

4.1.1 Structural Analysis Model

A simple aircraft wing which has a NACA0012 airfoil profile is modeled parametrically in Catia V5-R16. The wing's three dimensional geometric model consists of 90 skin panels, 10 ribs and 4 spars while some of the skin panels are stiffened by cylindirical annular stringers along the wing span. The wing has a rectangular platform with 6m semi-span and 1.6m chord length. Finite element model of the wing is prepared by using Abaqus 6.7.1 and is composed of linear shell and beam elements. The model is shown in Figure 4.1, and consists of 17,070 linear quadrilateral elements of shell type, 1264 linear line elements of beam type, a total of 18,334 elements and 16,024 nodes, thus 96,144 degrees of freedom.

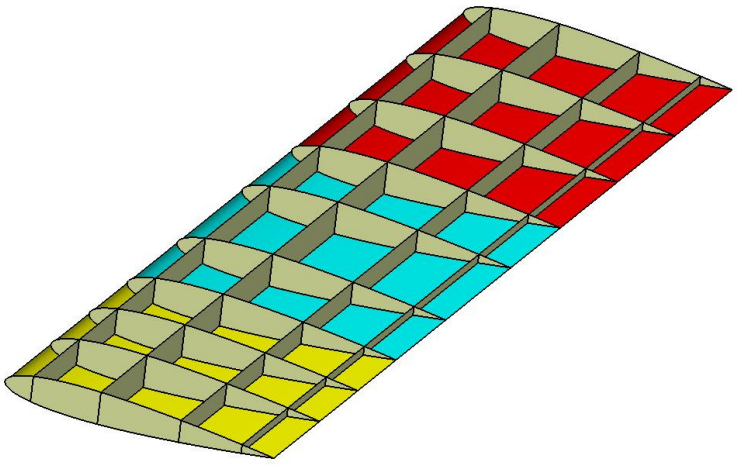


Figure 4.1 : Computational model of the wing structure

All members of the structure are made of aluminium with Young's modulus $E = 70000MPa$, Poisson ratio $\upsilon = 0.33$, density $\rho = 2700kg/m^3$, yield strength $\sigma_{yield} = 400MPa$. As a cantilevered boundary condition, all of the degrees of freedom at the root of the wing are set to zero. The aerodynamic load that will be applied to the wing is supplied from a computational fluid dynamics (CFD) analysis performed for the initial design. An Euler inviscid flow analysis was performed for $Mach = 0.3$ at sea level. For the sake of simplicity, the obtained total lift force of approximately $25000N$ is then expressed as an elliptic lift function (4.1) which changes along the wing span but assumed to be constant along the chord [48].

$$P(y) = (2.21*10^{-3})\sqrt{\left(1-\frac{y^2}{a^2}\right)} \qquad \textbf{(4.1)}$$

where a is the semi-span (6000 mm) and y is the point along the span on which the load value should be found in MPa. A static load analysis of this wing will be used as a reference to dictate the desired optimization constraints for this study. The structural criteria related to the reference analysis are shown in Table 4.1.

Table 4.1 : Structural analysis results of the reference wing

Criteria	Values
Maximum displacement	187 mm
Maximum Von Mises stress	202 MPa
Mass	336 kg
First modal frequency	4.35 Hz

4.1.2 Definition of Optimization Variables

Since ribs, spars and skin panels are modeled as shell elements, the thicknesses of these elements and the diameter of the stringers are chosen as design parameters. The thicknesses of spars, ribs and skin panels are divided into three groups along the wing span (as shown in Figure 4.1), introducing 9 design variables. The outer diameter of all the stringers are kept constant along the span and expressed as only one design parameter while the wall thickness of the stringers are taken as one over third of the outer diameter. In Figure 4.1, the structural components of the wing and the thickness parameters related to these components are presented so that each different color shows a different design parameter.

Since the wing structure is divided into three sections along its span, there are three independent skin panel thickness variables (t_{A1}, t_{A2}, t_{A3}), three independent spar thickness variables (t_{B1}, t_{B2}, t_{B3}) and three independent rib thickness variables (t_{C1}, t_{C2}, t_{C3}) in the wing structure. In Figure 4.1, each different color shows a different independent thickness parameter. Each of these thicknesses is related to its initial thickness value $\tilde{t}_i$ through an optimization parameter representing the percentile change in thickness n_i. n_{A1} is the percentile change in the thicknesses of skin panel in first section namely cantilevered side, n_{A2} is the percentile change in the thicknesses of skin panel in the middle section, and n_{A3} is the percentile change in the thicknesses of skin panel in the tip section. Thus, skin panel thicknesses are allowed to change as follows;

$$t_{A1} = n_{A1}\tilde{t}_{A1} \qquad t_{A2} = n_{A2}\tilde{t}_{A2} \qquad t_{A3} = n_{A3}\tilde{t}_{A3} \tag{4.2}$$

where, t_{A1}, t_{A2}, t_{A3} are the physical design variables describing the skin panel thicknesses for the three partitions along the span. t_{A1} is chosen to be on the cantilevered side. $\tilde{t}_{A1}, \tilde{t}_{A2}, \tilde{t}_{A3}$ are the initial values for the thicknesses of skin panels in three sections. Thus, spar thicknesses are allowed to change as follows;

$$t_{B1} = n_{B1}\tilde{t}_{B1} \qquad t_{B2} = n_{B2}\tilde{t}_{B2} \qquad t_{B3} = n_{B3}\tilde{t}_{B3} \tag{4.3}$$

where, t_{B1}, t_{B2}, t_{B3} are the physical design variables describing the spar thicknesses for the three partitions along the span. t_{B1} is chosen to be on the cantilevered side. $\tilde{t}_{B1}, \tilde{t}_{B2}, \tilde{t}_{B3}$ are the initial values for the thicknesses of the three spar partitions. Finally, rib thicknesses are allowed to change as follows;

$$t_{C1} = n_{C1}\tilde{t}_{C1} \qquad t_{C2} = n_{C2}\tilde{t}_{C2} \qquad t_{C3} = n_{C3}\tilde{t}_{C3} \tag{4.4}$$

where, t_{C1}, t_{C2}, t_{C3} are the physical design variables describing the rib thicknesses for the three partitions along the span. t_{C1} is chosen for the first rib on the cantilevered side. $\tilde{t}_{C1}, \tilde{t}_{C2}, \tilde{t}_{C3}$ are the initial values for the thicknesses of the three different rib groups.

The outer diameter of all the stringers are kept constant along the span and expressed as only one design parameter while the wall thickness of the stringers are taken as one over third of the outer diameter. Thus, two more design variables, the stringer outer diameter d_0 and the inner wall thickness of the stringer beam t_w are:

$$d_0 = n_{10}\bar{d}_0 \qquad t_w = \frac{d_0}{3} \tag{4.5}$$

where d_0 is the reference diameter value of the initial wing design.

The thickness optimization variables are constrained to be less than one so that the initial bulk structure will get lighter with respect to the initial weight. The lower and upper limits of the thickness optimization variables are chosen as:

$$0.192 \le n_{A1} \le 1.0 \qquad 0.048 \le n_{A2} \le 1.0 \qquad 0.064 \le n_{A3} \le 1.0 \tag{4.6}$$

$$0.640 \le n_{B1} \le 1.0 \qquad 0.480 \le n_{B2} \le 1.0 \qquad 0.320 \le n_{B3} \le 1.0$$

$$0.480 \le n_{C1} \le 1.0 \qquad 0.240 \le n_{C2} \le 1.0 \qquad 0.080 \le n_{C3} \le 1.0$$

$$0.2 \le n_{10} \le 1.0$$

In addition, the location of the first four ribs which is the group on the wing root side and also the location of the middle two spars are variables. The absolute distances from the root to each of the first four ribs are chosen as four optimization variables y_1, y_2, y_3, y_4.

$$500mm \le y_1 \le 800mm \tag{4.7}$$

$$900mm \le y_2 \le 1300mm$$

$$1400mm \le y_3 \le 1950mm$$

$$2150mm \le y_4 \le 2800mm$$

For two middle spars, the ratio of the distance between the leading edge of the wing to the spar divided by the chord length is chosen as two dimensionless optimization variables c_1, c_2.

$$0.25 \le c_1 \le 0.45 \tag{4.8}$$

$$0.55 \le c_2 \le 0.75$$

A rather bulk wing initial design will be given for the optimization problem since optimization variables are chosen such as to reduce the thicknesses in any ways. At the initial configuration, $\tilde{t}_{A1} = \tilde{t}_{A2} = \tilde{t}_{A3} = 5mm$, $\tilde{t}_{B1} = \tilde{t}_{B2} = \tilde{t}_{B3} = 20mm$, $\tilde{t}_{C1} = \tilde{t}_{C2} = \tilde{t}_{C3} = 16mm$, $y1 = 600mm$, $y_2 = 1100mm$, $y_3 = 1600mm$, $y_4 = 2250mm$, $c_1 = 0.35$, $c_2 = 0.65$.

4.1.3 Deterministic Optimization of a Generic Aircraft Wing

The deterministic optimization that will be solved has two objectives as minimization of weight and maximization of the first modal frequency of the structure while constraining maximum Von Mises stress with the yield strength of the material. Yield strength is 400 MPa. A safety factor of 1.5 is used on the stress constraint in the deterministic optimization. The multi-objective optimization problem is formulated as;

$$\min_{s \in S} M(\boldsymbol{s}), \max_{s \in S} f_1(\boldsymbol{s})$$

$$g_1(\boldsymbol{s}) = \frac{\sigma_{yield}}{1.5 * \sigma_{max}(\boldsymbol{s})} - 1 \geq 0, \qquad g_1(\boldsymbol{s}) \in \mathbb{R}$$

$$g_2(\boldsymbol{s}) = \frac{u_0}{u_{max}(\boldsymbol{s})} - 1 \geq 0, \qquad g_2(\boldsymbol{s}) \in \mathbb{R}$$

$$g_3(\boldsymbol{s}) = 1 - \frac{f_1^0}{f_1(\boldsymbol{s})} \geq 0, \qquad g_3(\boldsymbol{s}) \in \mathbb{R}$$

$$g_4(\boldsymbol{s}) = \frac{M_0}{M(\boldsymbol{s})} - 1 \geq 0, \qquad g_4(\boldsymbol{s}) \in \mathbb{R}$$

$$\boldsymbol{S} = \{\boldsymbol{s} \in \mathbb{R} | \boldsymbol{s_L} \leq \boldsymbol{s} \leq \boldsymbol{s_U}\} \tag{4.9}$$

where $M(\boldsymbol{s})$ is the total mass, $g_i(\boldsymbol{s})$ are the constraints, $u_{max}(\boldsymbol{s})$ and $\sigma_{max}(\boldsymbol{s})$ are the maximum tip displacement and maximum Von Mises stress of the wing structure. $u_0 = 187mm$ and $M_0 = 330kg$ are chosen as reference values from a reference wing to constrain the displacement and mass. $f_1(\boldsymbol{s})$ is the first natural frequency of the structure, while $f_1^0 = 4.35Hz$ is the first natural frequency of the reference wing. $\boldsymbol{S}$ is the set of optimization parameters with lower bound $\boldsymbol{s_L}$ and upper bound $\boldsymbol{s_U}$.

The designs which are found previously in the deterministic optimization process in [48] are given in Table 4.2. The design which corresponds to Pareto 1 in Table 4.2 is chosen as optimum design due to its minimum mass value while still satisfying constraints. Pareto 1 has a safety index of $\beta_{Deterministic} = 5.1492$.

Table 4.2 : Pareto designs of deterministic design of generic aircraft wing

Pareto	$\sigma_{\max}$ (MPa)	$u_{\max}$ (mm)	$Frequency$ (Hz)	$Mass$ (kg)
1	198.01	175.23	5.72	263.44
2	183.33	177.21	5.80	277.33
3	177.89	146.02	6.77	289.06

4.1.4 Reliability Based Design Optimization of a Generic Aircraft Wing

The reliability based design optimization problem that will be solved with two random variables (which are yield strength σ_{yield} and Young's modulus E of the material), has two objectives which are minimization of weight and maximization of the first modal frequency of the structure (Equation (4.10)). Thus, the constraints concerning stress (g_1), displacement (g_2) and frequency (g_3) in this problem become probabilistic constraints due to their dependencies on the random variables vector $\boldsymbol{X} = [\sigma_{yield} \;\; E]$. σ_{yield} and E are modeled with normal distributions assuming $N(400, 20)\ MPa$ and $N(70000, 350)\ MPa$ respectively. Then, the multi-objective optimization problem is formulated as;

$$\min_{s \in S} M(\boldsymbol{s}), \max_{s \in S} f_1(\boldsymbol{s})$$

$$P\left[g_1^{prob}(\boldsymbol{X}, \boldsymbol{s}) = \frac{\sigma_{yield}(\boldsymbol{X})}{\sigma_{max}(\boldsymbol{s})} - 1 \geq 0\right] \geq 1.0 - 10^{-7}, \qquad g_1^{prob}(\boldsymbol{X}, \boldsymbol{s}) \in \mathbb{R}$$

$$P\left[g_2^{prob}(\boldsymbol{X}, \boldsymbol{s}) = \frac{u_0}{u_{max}(\boldsymbol{X}, \boldsymbol{s})} - 1 \geq 0\right] \geq 1.0 - 10^{-7}, \qquad g_2^{prob}(\boldsymbol{X}, \boldsymbol{s}) \in \mathbb{R}$$

$$P\left[g_3^{prob}(\boldsymbol{X}, \boldsymbol{s}) = 1 - \frac{f_1^0}{f_1(\boldsymbol{X}, \boldsymbol{s})} \geq 0\right] \geq 1.0 - 10^{-7}, \qquad g_3{}^{prob}(\boldsymbol{X}, \boldsymbol{s}) \in \mathbb{R}$$

$$g_4^{det}(\boldsymbol{s}) = \frac{M_0}{M(\boldsymbol{s})} - 1 \geq 0, \qquad g_4{}^{det}(\boldsymbol{s}) \in \mathbb{R}$$

$$\boldsymbol{S} = \{\boldsymbol{s} \in \mathbb{R} | \boldsymbol{s_L} \leq \boldsymbol{s} \leq \boldsymbol{s_U}\} \qquad \textbf{(4.10)}$$

where $M(\boldsymbol{s})$ is the total mass, $g_i^{prob}(\boldsymbol{X}, \boldsymbol{s})$ are the probabilistic constraints, $g_i^{det}(\boldsymbol{s})$ are the deterministic constraints, $u_{max}(\boldsymbol{X}, \boldsymbol{s})$ and $\sigma_{max}(\boldsymbol{s})$ are the maximum tip displacement and maximum Von Mises stress of the wing structure. $u_0 = 187mm$ and $M_0 = 330kg$ are chosen as reference values from a reference wing to constrain the displacement and mass. $f_1(\boldsymbol{X}, \boldsymbol{s})$ is the first natural frequency of the structure, while $f_1^0 = 4.35Hz$ is the first natural frequency of the reference wing. $\boldsymbol{S}$ is the set of

optimization parameters with lower bound $\boldsymbol{s_L}$ and upper bound $\boldsymbol{s_U}$. After some computations, the frequency constraint is not considered to be dominant in the reliability analysis and treated as a deterministic constraint for the sake of simplicity. In terms of reliability index, the above optimization problem can be expressed as;

$$\min_{s \in S} M(\boldsymbol{s}), \max_{s \in S} f_1(\boldsymbol{s})$$

$$g_1^{prob}(\beta_{Stress}) = \beta_{Stress} - \beta_{Target\ Stress} \geq 0, \qquad g_1{}^{prob}(\beta_{Stress}) \in \mathbb{R}$$

$$g_2^{prob}(\beta_{Disp}) = \beta_{Disp} - \beta_{Target\ Disp} \geq 0, \qquad g_2{}^{prob}(\beta_{Disp}) \in \mathbb{R}$$

$$g_3^{det}(\boldsymbol{s}) = 1 - \frac{f_1^0}{f_1(\boldsymbol{s})} \geq 0, \qquad g_3{}^{det}(\boldsymbol{s}) \in \mathbb{R}$$

$$g_4^{det}(\boldsymbol{s}) = \frac{M_0}{M(\boldsymbol{s})} - 1 \geq 0, \qquad g_4{}^{det}(\boldsymbol{s}) \in \mathbb{R}$$

$$\boldsymbol{S} = \{\boldsymbol{s} \in \mathbb{R} | \boldsymbol{s_L} \leq \boldsymbol{s} \leq \boldsymbol{s_U}\} \qquad \textbf{(4.11)}$$

Here, $\beta_{Target\ Stress}$ and $\beta_{Target\ Disp}$ are the target reliability indexes for stress and displacement constraints and chosen as to be 5.1993 for a probability of faiure of 10^{-7}. They could have different values but here the same reliability level is chosen. Equation (4.11) is solved using Modefrontier as an optimizer driver. FORM code written in Matlab is used for reliability analysis. In Appendix A.8, FORM code with the script files written in Matlab is shown. Catia is used as a parametric solid modeler while Abaqus is used to compute the structural response of the wing system. The optimization workflow for reliability based design optimization (RBDO) is shown in Figure 4.2.

Solution of the present study took about 72 hours 30 minutes using Modefrontier on a workstation with Intel(R) Core(TM) 2 Quad CPU Q8300@2.50 GHz processor, with 2.00 GB of RAM on Microsoft Windows XP operating system. At least twice of the number of optimization parameters should be given as the number of design of experiments (DOE) (as stated in Modefrontier's user manual). So, in this study, 52 DOE is used. "Sobol Sequence" is employed to distribute DOE points because Modefrontier's user manual recommended "Sobol Sequence" as initial design population for MOGA algorithm. MOGA-II (Multi Objective Genetic Algorithm II) is used for attaining optimum result. Finally a total number of 290 designs are generated for the optimization problem. As a result 75 unfeasible designs and 215 feasible designs are found. The values of pareto designs are given in Table 4.3.

Table 4.3 : Pareto designs of reliability based structural design

Variables & Criteria	1	2	3	4	5	6	7
nA1 (mm)	0.85	0.92	0.92	0.92	0.82	0.92	0.92
nA2 (mm)	0.40	0.30	0.25	0.27	0.40	0.25	0.27
nA3 (mm)	0.14	0.14	0.17	0.17	0.14	0.17	0.17
nB1 (mm)	0.85	0.82	0.69	0.85	0.85	0.69	0.85
nB2 (mm)	0.68	0.95	0.73	0.63	0.63	0.73	0.63
nB3 (mm)	0.40	0.35	0.50	0.52	0.52	0.50	0.52
nC1 (mm)	0.68	0.83	0.55	0.50	0.50	0.55	0.50
nC2 (mm)	0.44	0.52	0.24	0.24	0.24	0.24	0.24
nC3 (mm)	0.23	0.15	0.25	0.23	0.23	0.25	0.23
nD1 (mm)	0.40	0.33	0.43	0.38	0.38	0.43	0.28
rib1_ref	16	16	16	16	16	16	16
rib2_ref	16	16	16	16	16	16	16
rib3_ref	16	16	16	16	16	16	16
rib_pos_1 (mm)	550	550	700	700	650	700	600
rib_pos_2 (mm)	950	950	1100	1100	1100	1100	1000
rib_pos_3 (mm)	1500	1450	1650	1900	1600	1900	1500
rib_pos_4 (mm)	2250	2250	2650	2750	2500	2750	2400
skin1_ref	5	5	5	5	5	5	5
skin2_ref	5	5	5	5	5	5	5
skin3_ref	5	5	5	5	5	5	5
spar1_ref	20	20	20	20	20	20	20
spar2_ref	20	20	20	20	20	20	20
spar3_ref	20	20	20	20	20	20	20
spar_pos_1 (mm)	0.25	0.25	0.45	0.45	0.40	0.45	0.25
spar_pos_2 (mm)	0.55	0.55	0.75	0.55	0.60	0.75	0.55
stringer_outer_ref	15	15	15	15	15	15	15
β_{Stress}	12.33	11.15	10.66	11.84	11.74	11.85	9.09
σ_{yield} (MPa)	153.36	177.04	186.90	163.21	165.21	163.07	218.23
β_{Disp}	30.25	25.93	6.19	32.32	26.86	18.89	12.48
E (MPa)	59413	60925	67832	58687	60600	63388	65631
Frequency (Hz)	6.761	7.014	6.126	6.419	6.404	6.333	6.136
Mass (kg)	268	278	246	267	258	253	253
Improvement of Mass	+%1.9	+%5.7	-%6.5	+%1.5	-%1.9	-%3.8	-%3.8
Improvement of Frequency	+%18	+%22	+%7	+%12	+%11	+%10	+%7

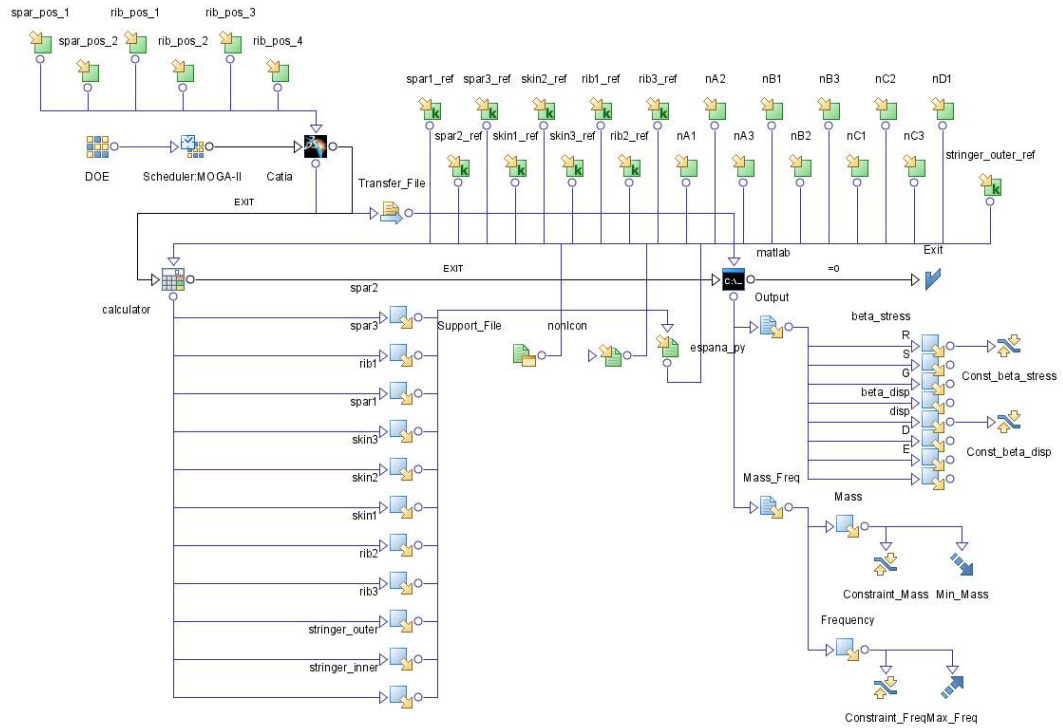


Figure 4.2 : Modefrontier workflow for reliability based structural design

According to the desired criteria, one of the paretos can be chosen as the most preferred design. The design which corresponds to Pareto 3 in Table 4.3 is chosen as optimum design due to its minimum mass value while still satisfying the target reliability index constraints.

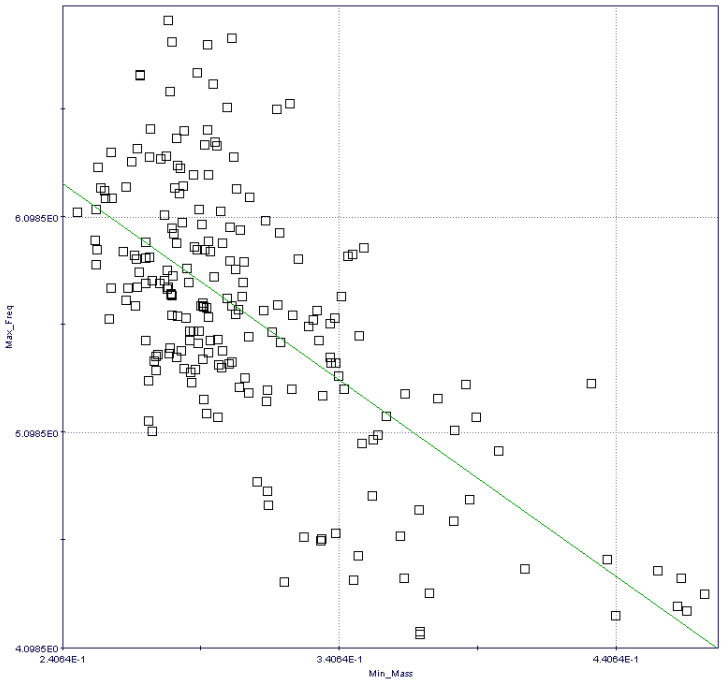


Figure 4.3 : Mass vs frequency space

The selected pareto design 3 gives approximately %6.5 decrease in mass and %7 increase in frequency with respect to the reference values of 263 kg and 5.72 Hz. All feasible designs are shown in mass vs frequency space in Figure 4.3 where the regression line which shows the relationship between objective functions for the feasible design points are demonstrated. As it is seen from the regression line in Figure 4.3, the frequency values are decreasing while mass values are increasing as expected.

In reliability based design optimization of a generic aircraft wing study, the pareto designs' reliability constraints are over-satisfied, other optimization drivers except Modefrontier may prevent this issue.

4.2 Computational Aeroelastic Analysis

A realistic aeroelastic analysis is based on coupling of structural and aerodynamic responses. In this present study, Mpcci (Mesh-based Parallel Code Coupling Interface) is used for this required aeroelastic coupling interface.

While Abaqus-6.7.1 is used as finite element solver, 3D inviscid Euler equations are solved with Fluent-6.3.26. The related two softwares are coupled by the use of Mpcci to satisfy the required conditions of structural and aerodynamic responses in an aeroelastic analysis. Then, Abaqus finds the displacements by using the aerodynamic loads calculated with Fluent.

For a general aeroelastic system consisted of mass-damping-spring model, the equation of motion can be written in an implicit form as below:

$$[M]\{\ddot{u}\}+[D]\{\dot{u}\}+[K]\{u\}=\{F_a\}+\{F_e\} \qquad \textbf{(4.12)}$$

We can ignore the time dependent terms in above equation since only the static aeroelastic effects are considered in this study. Moreover, the aeroelastic analysis includes only aerodynamic effects for force definition. By using these assumptions, we can derive an equation which is also compatible with finite element analysis technique:

$$[K]\{u\}=\{F_a\} \qquad \textbf{(4.13)}$$

In this study, the flow was assumed to be inviscid while Euler equations were used to model and solve it. 3D inviscid Euler equations can be written in a conservative form:

$$\frac{\partial w}{\partial t} + \vec{\nabla} \bullet \vec{F}(w) = 0 \tag{4.14}$$

where w is conservative fluid state variable and can be defined as:

$$w = \begin{Bmatrix} \rho \\ \rho u_1 \\ \rho u_2 \\ \rho u_3 \\ E \end{Bmatrix} \tag{4.15}$$

where ρ denotes the density of fluid; u_1, u_2, u_3 are velocities and E is the total internal energy per mass.

Flux can also be represented by using three components $\vec{F}_1, \vec{F}_2, \vec{F}_3$ as:

$$\vec{F}_1 = \begin{Bmatrix} \rho u_1 \\ \rho u_1^2 + p \\ \rho u_1 u_2 \\ \rho u_1 u_3 \\ (E+p)u_1 \end{Bmatrix}, \vec{F}_2 = \begin{Bmatrix} \rho u_2 \\ \rho u_2 u_1 \\ \rho u_2^2 + p \\ \rho u_2 u_3 \\ (E+p)u_2 \end{Bmatrix}, \vec{F}_1 = \begin{Bmatrix} \rho u_3 \\ \rho u_3 u_1 \\ \rho u_3 u_2 \\ \rho u_3^2 + p \\ (E+p)u_3 \end{Bmatrix} \tag{4.16}$$

4.3 AGARD 445.6 Wing Introduction

In the final step of this study, RBDO is applied to an aeroelastic optimization problem. The well-known AGARD (Advisory Group for Aerospace Research and Development) 445.6 wing is chosen as the wing structure. This wing is the first aeroelastic configuration that is tested by Yates [54] in the "Transonic Dynamics Tunnel (TDT)" at the NASA Langley Research Center. Deterministic aeroelastic optimization of AGARD 445.6 wing problem was presented formerly in the work of Nikbay [47] for the free stream Mach number of 0.85 and the angle of attack of 5 degrees. In the present work, we consider a set of random variables for fluid and structural domain $\boldsymbol{X} = [\sigma_{yield}\ M\ \alpha]$ while uncertainties in yield strength, Mach number and angle of attack will be accounted in the reliability analysis.

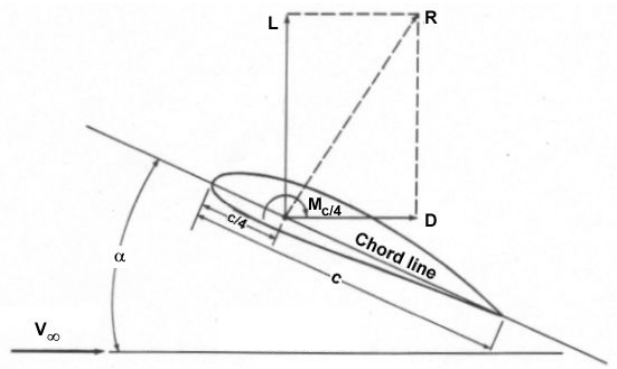
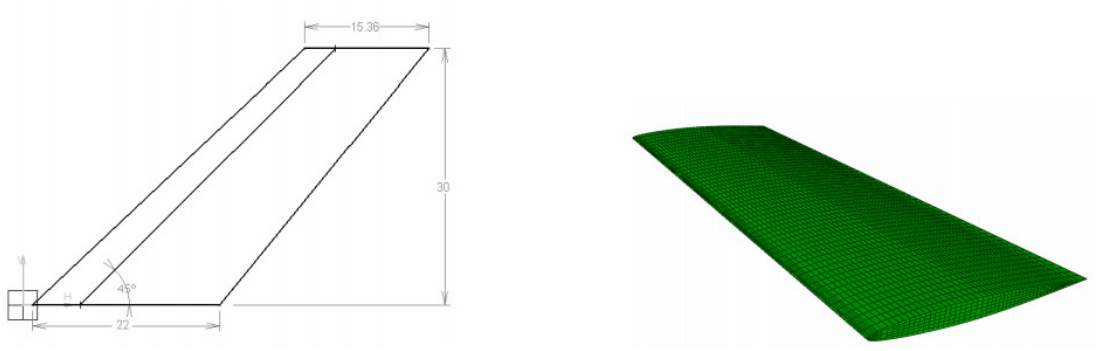


Figure 4.4 : Definition of angle of attack on an airfoil, lift and drag forces

4.3.1 Aeroelastic Analysis Model

The AGARD 445.6 wing is a swept-back wing with a quarter-chord sweep angle of 45 degrees. Cross sections of the wing are NACA 65A004 airfoils. The wing has a taper ratio of 0.66 and an aspect ratio of 1.65. Moreover, it is a wall-mounted model made with laminated mahogany. The wings parametric CAD model prepared with Catia-V5 is given in Figure 4.5. There are 2 models of the AGARD 445.6 wing: solid and weakened model. In this study weakened model of the wing is used. The finite element model in Abaqus is composed of 19,610 linear hexahedral structural elements. The computational grid of the flow domain was constructed in Gambit with 691,000 tetrahedral elements and 1.35 million faces. The flow is modeled with the Euler equations.



(a) Wing geometry (lengths in inches)

(b) The finite element model of the wing

Figure 4.5 : AGARD 445.6 wing geometry and FEM model

4.3.2 Deterministic Optimization of AGARD 445.6 Wing

There are two objective functions in this deterministic optimization problem which are maximizing the $\frac{L}{D}$ ratio and minimizing the weight (Equation (4.17)). Mach

number and angle of attack are 0.85 and 5 respectively. Then, the multi-objective optimization problem is formulated as;

$$\min_{s \in S} M(\boldsymbol{s}), \max_{s \in S} \frac{L}{D}(\boldsymbol{s})$$

$$g_1(\boldsymbol{s}) = \frac{u_0}{u_{max}(\boldsymbol{s})} - 1 \geq 0, \qquad g_1(\boldsymbol{s}) \in \mathbb{R}$$

$$\boldsymbol{S} = \{\boldsymbol{s} \in \mathbb{R} | \boldsymbol{s_L} \leq \boldsymbol{s} \leq \boldsymbol{s_U}\} \quad \boldsymbol{s} = \left(\lambda, \Lambda_{\frac{c}{4}}\right) \; 0.1 \leq \lambda \leq 0.5 \;\; 0^\circ \leq \Lambda_{\frac{c}{4}} \leq 50^\circ \tag{4.17}$$

where $M(\boldsymbol{s})$ is the total mass of the wing, $\frac{L}{D}(\boldsymbol{s})$ is the lift over drag value for the wing. $g_i(\boldsymbol{s})$ are the constraints, $u_{max}(\boldsymbol{s})$ is the maximum tip displacement. λ is the taper ratio defined as $\lambda = \frac{c_{tip}}{c_{root}}$ and $\Lambda_{\frac{c}{4}}$ is the sweep value at the quarter chord. $u_0 = 76mm$ is chosen as reference value to constrain the displacement. $\boldsymbol{S}$ is the set of optimization parameters with lower bound $\boldsymbol{s_L}$ and upper bound $\boldsymbol{s_U}$.

The designs which are found previously in the deterministic optimization process in [47] are given in Table 4.4. The design which corresponds to Pareto 3 in Table 4.4 is chosen as optimum design due to its minimum mass value while still satisfying the displacement constraint.

Table 4.4 : Pareto designs of deterministic design of AGARD 445.6 wing

Pareto	$u_{\max}$ (mm)	$\frac{L}{D}$	$Mass$ (kg)
1	60.6254	12.5754	1.4885
2	70.6557	12.2043	1.0853
3	57.9876	11.4589	0.9715

4.3.3 Reliability Based Aeroelastic Optimization of AGARD 445.6 Wing

There are two objective functions in this problem which are maximizing the $\frac{L}{D}$ ratio and minimizing the weight (Equation (4.18)). The yield strength σ_{yield}, free stream Mach number M and the angle of attack α are modeled with normal distributions assuming $N(8, 0.4)\, MPa$, $N(0.85, 0.03)$ and $N(5, 0.25)$ respectively. Then, the multi-objective optimization problem is formulated as;

$$\min_{s \in S} M(\boldsymbol{s}), \max_{s \in S} \frac{L}{D}(\boldsymbol{s})$$

$$P\left[\, g_1^{prob}(\boldsymbol{X},\boldsymbol{s}) = \frac{\sigma_{yield}(\boldsymbol{X})}{\sigma_{max}(\boldsymbol{X},\boldsymbol{s})} - 1 \;\geq 0\right] \geq 1.0 - 10^{-7}, \quad g_1^{prob}(\boldsymbol{X},\boldsymbol{s}) \in \mathbb{R}$$

$$P\left[g_2^{prob}(\boldsymbol{X},\boldsymbol{s}) = \frac{u_0}{u_{max}(\boldsymbol{X},\boldsymbol{s})} - 1 \;\geq 0\right] \geq 1.0 - 10^{-7}, \quad g_2^{prob}(\boldsymbol{X},\boldsymbol{s}) \in \mathbb{R}$$

$$\boldsymbol{S} = \{\boldsymbol{s} \in \mathbb{R} | \boldsymbol{s_L} \leq \boldsymbol{s} \leq \boldsymbol{s_U}\} \quad \boldsymbol{s} = \left(\lambda, \Lambda_{\frac{c}{4}}\right) \;\; 0.1 \leq \lambda \leq 0.5 \;\; 0^\circ \leq \Lambda_{\frac{c}{4}} \leq 50^\circ \qquad \textbf{(4.18)}$$

where $M(\boldsymbol{s})$ is the total mass of the wing, $\frac{L}{D}(\boldsymbol{s})$ is the lift over drag value for the wing. $g_i^{prob}(\boldsymbol{X},\boldsymbol{s})$ are the probabilistic constraints, $u_{max}(\boldsymbol{X},\boldsymbol{s})$ and $\sigma_{max}(\boldsymbol{X},\boldsymbol{s})$ are the maximum tip displacement and maximum Von Mises stress of the wing structure. λ is the taper ratio defined as $\lambda = \frac{c_{tip}}{c_{root}}$ and $\Lambda_{\frac{c}{4}}$ is the sweep value at the quarter chord.

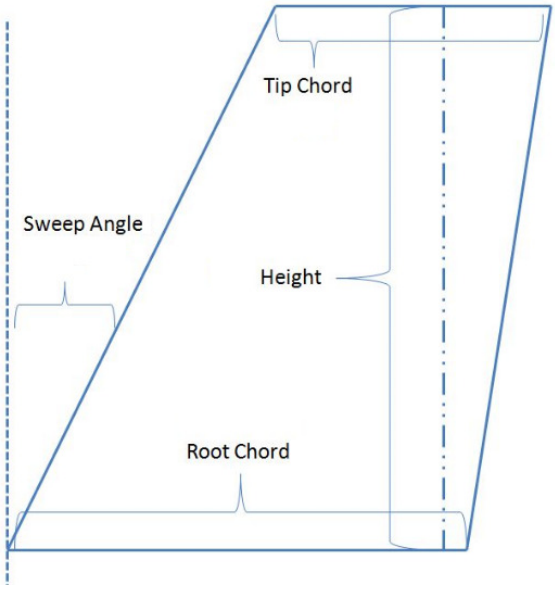


Figure 4.6 : Sweep angle, tip and root chords

$u_0 = 76mm$ is chosen as reference value to constrain the displacement. $\boldsymbol{S}$ is the set of optimization parameters with lower bound $\boldsymbol{s_L}$ and upper bound $\boldsymbol{s_U}$. In terms of reliability index, the above optimization problem can be expressed as;

$$\min_{s \in S} M(\boldsymbol{s}), \max_{s \in S} \frac{L}{D}(\boldsymbol{s})$$

$$g_1^{prob}(\beta_{Stress}) = \beta_{Stress} - \beta_{Target\ Stress} \geq 0, \quad g_1{}^{prob}(\beta_{Stress}) \in \mathbb{R}$$

$$g_2^{prob}(\beta_{Disp}) = \beta_{Disp} - \beta_{Target\ Disp} \geq 0, \quad g_2^{prob}(\beta_{Disp}) \in \mathbb{R}$$

$$\boldsymbol{S} = \{\boldsymbol{s} \in \mathbb{R} | \boldsymbol{s_L} \leq \boldsymbol{s} \leq \boldsymbol{s_U}\} \quad \boldsymbol{s} = \left(\lambda, \Lambda_{\frac{c}{4}}\right) \quad 0.1 \leq \lambda \leq 0.5 \quad 0^\circ \leq \Lambda_{\frac{c}{4}} \leq 50^\circ \tag{4.19}$$

Here, the target reliability indexes for stress and displacement constraints are chosen as to be 5.1993 for a probability of faiure of 10E-7. Several commercial software codes were coupled during the optimization process in this problem. Fluent-6.3.26 is used to solve inviscid 3D Euler equations, Gambit to generate the fluid domain mesh generator and Catia-V5-R16 to model the parametric 3D solid. Abaqus-6.7.1 was used to compute the structural response of the aeroelastic system. Mesh based parallel code coupling interface Mpcci-3.0.6 was used to exchange the pressure and displacement information between Fluent and Abaqus. Modefrontier-4.0 was used as a multi-objective and multidisciplinary optimization software to solve the Equation (4.19). FORM code written in Matlab is used for reliability analysis. The flowchart of the FORM code is given in Figure 4.7. In Appendix A.9, FORM code with the script files written in Matlab is shown.

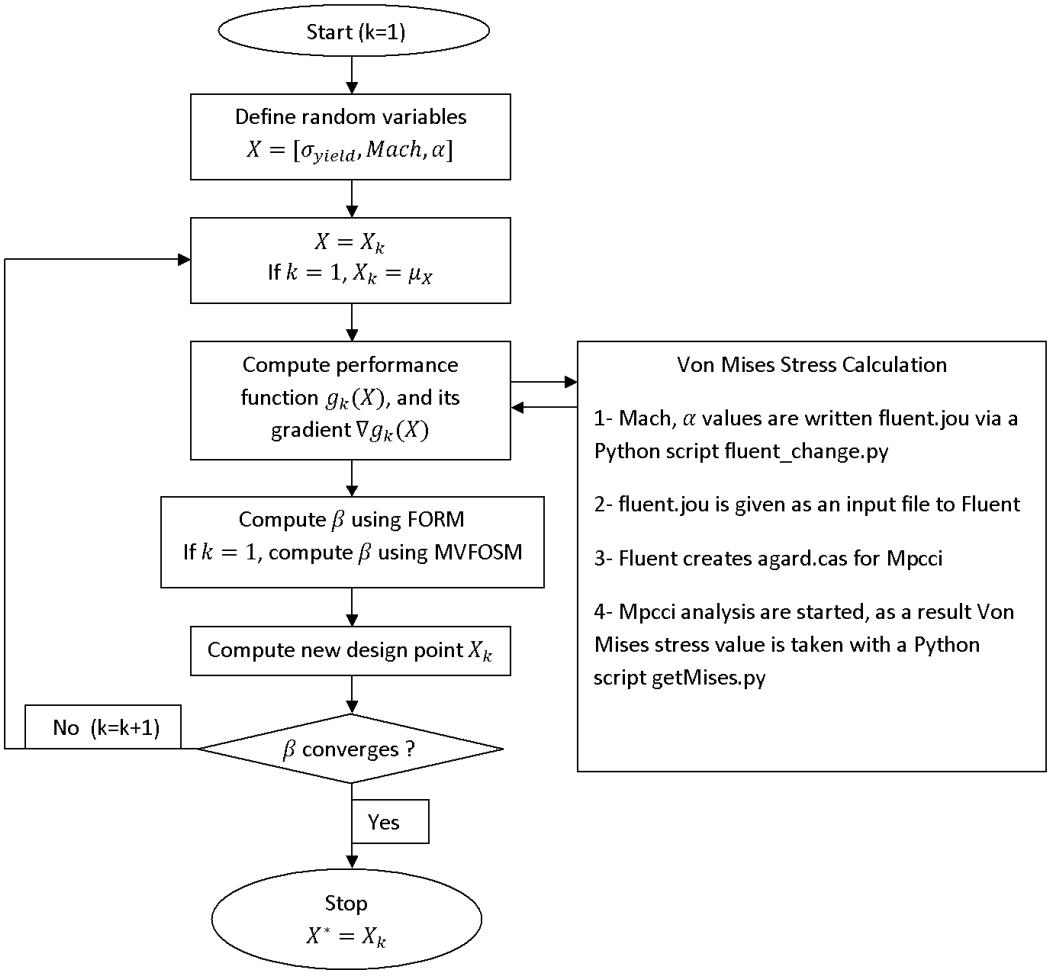


Figure 4.7 : Flowchart of the FORM code for AGARD 445.6 wing

The optimization workflow for reliability based aeroelastic design can be shown in Figure 4.8. In Figure 4.8, Catia V5 node updates the optimization variables by using the parametric 3D CAD model. Then, the new geometric model is transferred to Gambit in “iges” format. Gambit uses a journal file to prepare the fluid mesh and to update the boundary conditions and then transfers the mesh file to Fluent node. Fluent updates the optimization variables and imports the mesh files. Next, Fluent prepares the flow model and sets boundary conditions through a journal file and transfers the “case” file to Mpcci for the aeroelastic analysis. In CSD preprocessing, Catia V5 node updates the optimization variables by using the parametric 3D CAD model. Abaqus updates the structural model by using a Python script and transfers the input file to Mpcci for the aeroelastic analysis. Then Mpcci performs the coupling by using the Fluent and Abaqus models in batch mode. This aeroelastic analysis produces a result file that contains the aerodynamic and structural criteria. At each optimization iteration, an inner loop for reliability analysis is performed at the Matlab node. Modefrontier controls the constraint violation for both deterministic and probabilistic criteria. MOGA-II controls the optimization process and if needed, new iteration process starts.

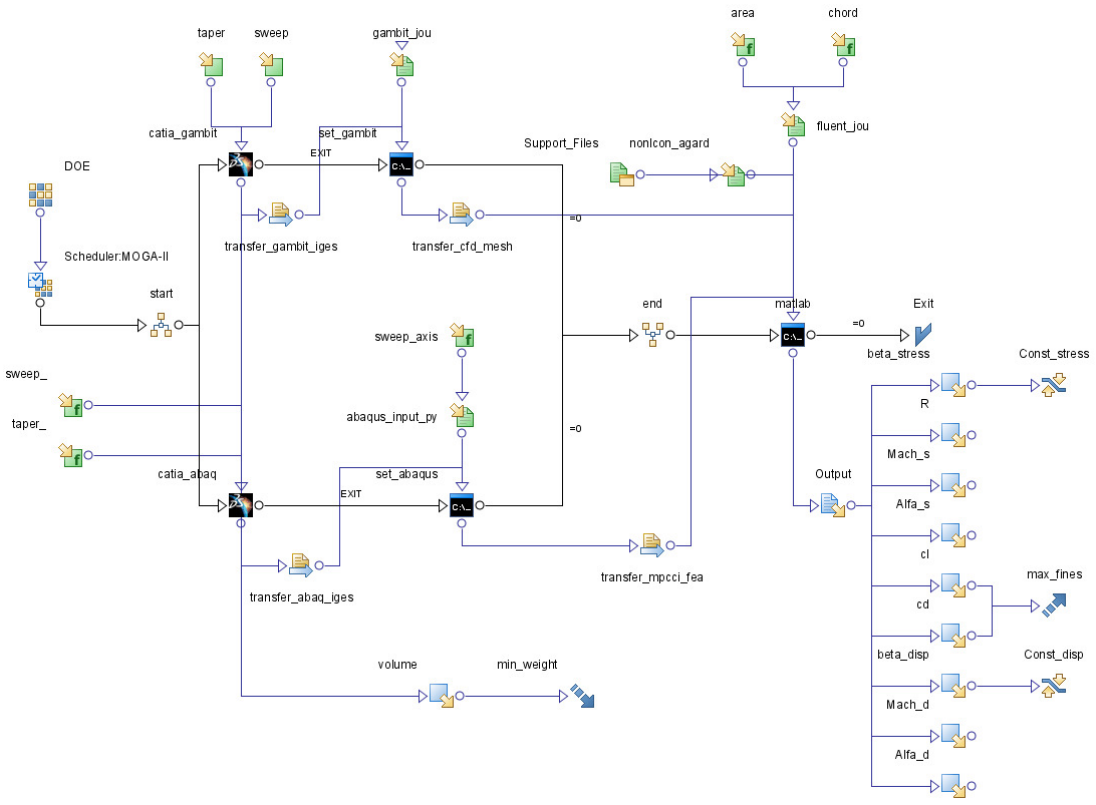


Figure 4.8 : Modefrontier workflow for reliability based aeroelastic design

Solution of the present study took about 215 hours 40 minutes using Modefrontier on a workstation with Intel(R) Core(TM) 2 Quad CPU Q6700@2.40 GHz processor, with 2.00 GB of RAM on Microsoft Windows XP operating system. In this study, 12 design of experiments (DOE) is used. "Sobol Sequence" is employed to distribute DOE points. MOGA-II (Multi Objective Genetic Algorithm II) is used for attaining optimum result. Finally a total number of 43 designs are generated for the optimization problem. As a result 20 unfeasible designs and 23 feasible designs are found. The values of pareto designs are given in Table 4.5. According to the desired criteria, one of the paretos can be chosen as the most preferred design. The design which corresponds to Pareto 3 in Table 4.5 is chosen as optimum design due to its minimum mass value, satisfactory $\frac{L}{D}$ ratio, while still satisfying the target reliability index constraints.

Table 4.5 : Pareto designs of reliability based aeroelastic design

Variables & Criteria	1	2	3	4	5	6
Sweep	8	16	8	10	10	18
Taper	0.225	0.275	0.200	0.225	0.250	0.325
C_D	0.039	0.038	0.040	0.039	0.039	0.037
C_L	0.454	0.447	0.457	0.454	0.451	0.438
β_{Stress}	6.133	5.444	6.323	6.045	5.858	5.233
σ_{yield} (MPa)	6.109	6.484	6.031	6.152	6.221	6.628
M_{Stress}	0.913	0.927	0.911	0.912	0.910	0.933
α_{Stress}	5.823	5.740	5.853	5.828	5.809	5.703
β_{Disp}	5.923	5.296	5.374	5.555	6.057	5.209
M_{Disp}	0.928	1.009	0.924	0.928	0.928	0.936
α_{Disp}	6.333	5.000	6.193	6.228	6.369	6.087
L/D	11.539	11.713	11.534	11.561	11.576	11.767
Mass (kg)	1.116	1.182	1.085	1.116	1.149	1.252
Improvement of Mass	+%14.9	+%21.0	+%11.7	+%14.9	+%18.3	+%28.9
Improvement of L/D	+%0.7	+%2.2	+%0.7	+%0.9	+%1.0	+%2.7

The selected pareto design 3 gives approximately %11.7 increase in mass and %0.7 increase in L/D ratio with respect to the reference values of 0.9715 kg and 11.4589. All feasible designs are shown in mass vs L/D ratio space in Figure 4.9 where the regression line which shows the relationship between objective functions for the feasible design points are demonstrated. As it is seen from the regression line in Figure 4.9, the L/D values are increasing while mass values are increasing.

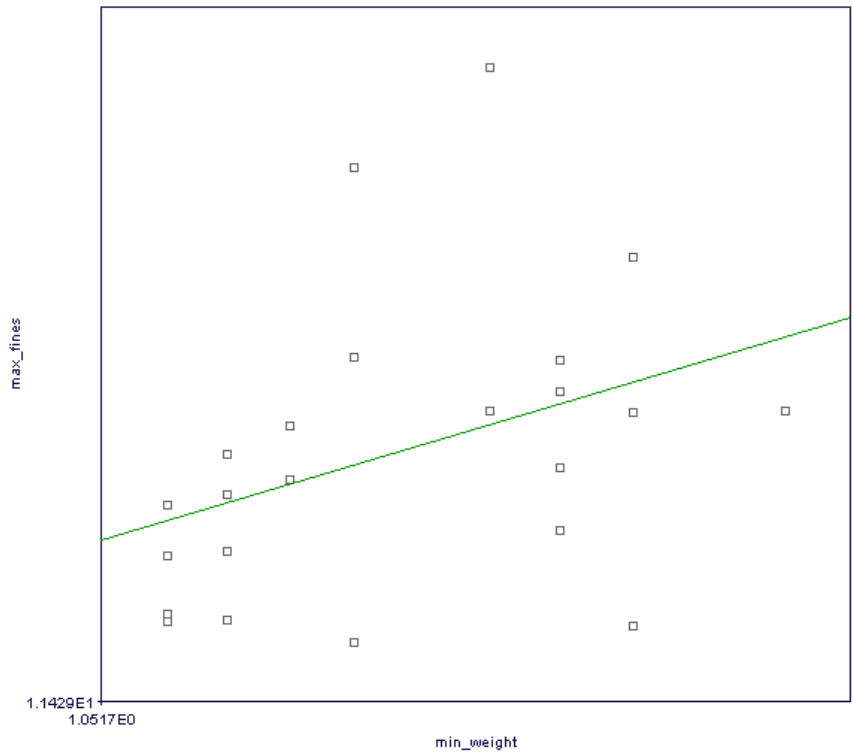


Figure 4.9 : Mass vs L/D ratio space

5. CONCLUSION AND FUTURE WORK

In this work, Reliability Index Approach (RIA) and Performance Measure Approach (PMA) are implemented in an in-house developed RBDO code and integrated into a multidisciplinary optimization framework composed of high-fidelity commercial softwares. In this framework, finite volume based flow solver Fluent is used to solve inviscid 3D Euler equations and Catia is used as a parametric 3D solid modeler. Abaqus, a structural finite element method solver, is used to compute the structural response of the aeroelastic system. MPCCI, mesh based parallel code coupling interface, is used to exchange the pressure and displacement information between Fluent and Abaqus to perform a loosely coupled aeroelastic analysis. Modefrontier is employed as a multi-objective and multi-disciplinary optimization driver to control the optimization work flow. The optimization criteria include both deterministic and probabilistic constraints with both structural and aerodynamic uncertainties.

The RBDO methodology is validated with an example from the literature, then extended to optimization of a generic wing structure and finally applied to an aeroelastic optimization problem for AGARD 445.6 wing. In the wing structures, Hasofer-Lind iteration method is implemented in Matlab to compute MPP (Most Probable failure Point) solution. In the final application, random variation in structural and aerodynamic parameters such as in yield strength, Mach number and angle of attack are considered.

The presented reliability based multidisciplinary optimization process is proven to be fully-automatic, modular and practical which could find potential applications in industrial problems.

Future work for this study, the developed RBDO methodology could be applied to dynamic fluid-structure interaction problems. FORM and SORM are adequate for the low variations of the limit-state function, in high variations, expansion methods such as polynomial chaos expansion (PCE) can be used and the number of uncertain parameters could be increased. Usage of Mpcci for aeroelastic analysis takes too

much computation time, other software applications can be used to overcome this issue.

REFERENCES

[1] **R. Rubinstein,** 1981: Simulation and the Monte Carlo Method, John Wiley & Sons, New York.

[2] **M. Kalos, P. Whitlock,** 1986: Monte Carlo Methods Basics, vol. I, John Wiley & Sons, New York.

[3] **H. Kahn,** 1956: Simulation and the Monte Carlo Method, John Wiley & Sons, New York.

[4] **M. Shinozuka,** 1983. Basic analysis of structural safety, J. Struct. Engrg. 109.

[5] **Y.-T. Wu, H. Millwater, T. Cruse,** 1990: Advanced probabilistic structural analysis method for implicit performance functions, AIAA J. 12.

[6] **Y.-T. Wu,** 1992: Reliability Technology, vol. 28, The American Society of Mechanical Engineers, New York.

[7] **R. Melchers,** 1990: Radial importance sampling for structural reliability, J. Engrg. Mech. 116 (1).

[8] **S.-H. Kim, K.-W. Ra,** 2000: Adaptive importance sampling method for the stochastic finite element analysis, in: 8th Joint Specialty Conference on Probabilistic Mechanics and Structural Reliability, South Bend, IN, ASCE.

[9] **R. Ghanem, P. Spanos,** 1991: Stochastic Finite Element: a Spectral Approach, Springer.

[10] **P. Bjerager,** 1990: On computational methods for structural reliability analysis, Struct. Safety 9 (2).

[11] **G. Schueller,** 2001: Computational stochastic mechanics—recent advances, Comput. Struct. 79.

[12] **M. Capinski, N. Cutland,** 1991: Stochastic Navier Stokes equations, Acta Appl. Math. 25.

[13] **J. Kim,** 2002: On the stochastic Euler equations in a two-dimensional domain, SIAM J. Math. Anal. 33.

[14] **D. Xiu, D. Lucor, C.-H. Su, G. Karniadakis,** 2002: Stochastic modeling of flow–structure interactions using generalized polynomial chaos, J. Fluids Engrg. 124.

[15] **C. Cornell,** 1969: A probability-based structural code, J. Amer. Concr. Inst. 66 (12).

[16] **H. Madsen, S. Krenk, N. Lind,** 1986: Methods of Structural Safety, Prentice-Hall.

[17] **A. Hasofer, N. Lind,** 1974: Exact and invariant second-moment code format, J. Engrg. Mech. 100.

[18] **M.S. Eldred, H. Agarwal, V.M. Perez, S.F. Wojtkiewicz, J.E. Renaud,** 2007: Investigation of Reliability Method Formulations in DAKOTA/UQ.

[19] **J. Tu, K. K. Choi, Y. H. Park,** 1999: A New Study on Reliability-Based Design Optimization, Journal of Mechanical Design, Vol. 121 / 557.

[20] **Eggert, R.,** 1991: "Quantifying Design Feasibility Using Probabilistic Feasibility Analysis", 17th Design Automation Conference presented at the 1991 ASME Design Technical Conferences, New York, NY, USA, 32: 235-240, ASME.

[21] **Parkinson, A., Sorensen, C., and Pourhassan, N.,** 1993: "General Approach for Robust Optimal Design", Journal of Mechanical Design, Vol. 115.

[22] **Hohenbichler, M., Gollwitzer, S., and Kruse, W.,** 1987: "New Light on First- and Second Order Reliability Methods", Structural Safety, Vol. 4.

[23] **Koyluoglu, H. U. and Nielsen, S. R. K.,** 1994: "New Approximations for SORM Integrals", Structural Safety, Vol. 13.

[24] **Chen, W. J., Allen, K., and Tsui, K.,** 1996: "Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors", Journal of Mechanical Design, Vol. 118.

[25] **Sues, R. H., Oakley, D. R., and Rhodes, G. S.,** 1995: "Multidisciplinary Stochastic Optimization", Anonymous ASCE, Vol. 2.

[26] **Koch, P. N., Simpson, T. W., and Allen, J. K.,** 1999: "Statistical Approximations for Multidisciplinary Design Optimization: The Problem of Size", Journal of Aircraft , Vol. 36.

[27] **Zou, T. and Mahadevan, S.,** 2006: "A Direct Decoupling Approach for Efficient Reliability-Based Design Optimization", Structural and Multidisciplinary Optimization, Vol. 31.

[28] **Liu, H., Chen, W., and Sheng, J.,** 2003: "Application of the Sequential Optimization and Reliability Assessment Method to Structural Design Problems", ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Anonymous American Society of Mechanical Engineers, Chicago, IL, United States , 2 A: 63-72, ASME.

[29] **Liu, P.L., and Kiureghian, A.D.,** 1991: ''Optimization Algorithms For Structural Reliability'', Struct. Safety, 9, pp. 161–177.

[30] **Wang, L.P., and Grandhi, R.V,** 1994: ''Efficient Safety Index Calculation For Structural Reliability Analysis'', Comput. Struct., pp. 103–111.

[31] **Wang, L.P., and Grandhi, R.V,** 1996: 'Safety Index Calculation Using Intervening Variables For Structural Reliability'', Comput. Struct., pp. 1139–1148.

[32] **W, Y.T,** 1994: ''Computational Methods for Efficient Structural Reliability and Reliability Sensitivity Analysis'', AIAA J., pp. 1717–1723.

[33] **Yang, R. and Gu, L.,** 2004: "Experience with Approximate Reliability-Based Optimization Methods", Structural and Multidisciplinary Optimization, Vol. 26, pp. 152-159.

[34] **Du, X. and Chen, W.,** 2004: "Sequential Optimization and Reliability Assessment Method for Efficient Probabilistic Design", Journal of Mechanical Design, Vol. 126.

[35] **Thanedar, P. B. and Kodiyalam, S.,** 1991: "Structural Optimization Using Probabilistic Constraints", Proceedings of the 32nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference, 8-10 April, New York, NY, USA.

[36] **Tu, J., Choi, K. K., and Park, Y. H.,** 2001: "Design Potential Method for Robust System Parameter Design", AIAA Journal, Vol. 39.

[37] **Chen, X., Hasselman, T. K., and Neill, D. J.,** 1997: "Reliability Based Structural Design Optimization for Practical Applications", Anonymous AIAA, Vol. 4.

[38] **Royset, J. O., Der Kiureghian, A., and Polak, E.,** 2001: "Reliability-Based Optimal Structural Design by the Decoupling Approach", Reliability Engineering and System Safety, Vol. 73

[39] **Youn, B., Choi, K., and Park, Y.,** 2003: "Hybrid Analysis Method for Reliability-Based Design Optimization", Journal of Mechanical Design, Vol. 125.

[40] **Shan, S. and Wang, G.** 2008: "Reliable Design Space and Complete Single-Loop Reliability-Based Design Optimization", Reliability Engineering and System Safety, 93:1218-1230.

[41] **Pettit, C. L.,** 2004: "Uncertainty Quantification in Aeroelasticity: Recent Results and Research Challenges", AIAA Journal of Aircraft, Vol. 41.

[42] **Allen, M. and Maute, K.,** 2004: "Reliability Based Design Optimization of Aeroelastic Structures", Structural and Multidisciplinary Optimization, Vol. 27.

[43] **Hosder, S. and Maddalena, L.,** 2009: "Non-Intrusive Polynomial Chaos for the Stochastic CFD Study of a Supersonic Pressure Probe", 47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, 5 - 8 Jan 2009, Orlando, Florida, AIAA 2009-1129.

[44] **Kwon, J.H., L. J. K. H. and Kwak, M.,** 2009: "Reliability of Aerodynamic Analysis Using a Moment Method", International Journal of Computational Fluid Dynamics, Vol. 23.

[45] **Chassaing, J.C., a. L. D.,** 2010: "Stochastic Investigation of Flows About Airfoils at Transonic Speeds", AIAA Journal , Vol. 48.

[46] **Missoum, S., D. C. and Beran, P.,** 2010: "Reliability Based Design Optimization of Nonlinear Aeroelasticity Problems", AIAA Journal of Aircraft, Vol. 47.

[47] **Nikbay, M., Oncu, L., and Aysan, A.,** 2009: "Multidisciplinary Code Coupling for Analysis and Optimization of Aeroelastic Systems", AIAA Journal of Aircraft , Vol. 46.

[48] **Nikbay, M., Yanangonul, A., Oncu, L., and Kocas, M.,** 2008: "Multi-objective and gradient based structural design optimization of an aircraft wing", Second International Conference on Multidisciplinary Design Optimization and Applications, 2 - 5 Sep 2008, Gijon, Spain, ASMDO.

[49] **Nikbay, Ulucenk Ç., Yanangönül A. and Aysan A.,** 2009: "Reliability-Based Multi-Objective Design Optimization of an Aircraft Wing Structure with Abstract Optimization Variables", 5th Ankara International Aerospace Conference, 17 - 19 August 2009, METU, Ankara, Turkey.

[50] **Choi, S.-K., G. R. and Canfield, R.,** 2007: Reliability-Based Structural Optimization, Springer.

[51] **Madsen, H.O., Krenk, S., and Lind, N.C.,** 1986: Methods of Structural Safety, Prentice-Hall, Englewood Cliffs, NJ.

[52] **Arora, J.S.,** 1989: Introduction to Optimum Design, McGraw-Hill, New York, NY.

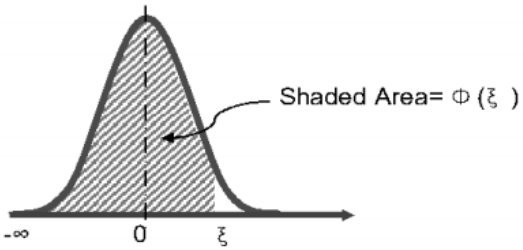
[53] **Sues, R., Aminpour, M., and Shin, Y.,** 2001: "Reliability-Based Multidisciplinary Optimization for Aerospace Systems", Proc. 42nd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 16-19 April, Seattle, WA, USA, AIAA.

[54] **Yates, E.,** 1985: AGARD Standard Aeroelastic Congurations for Dynamic Response I-Wing 445.6 , AGARD Report No.765.

APPENDICES

APPENDIX A.1

Cumulative Standard Normal Distribution Table



Statistical Table of Cumulative Standard Normal Distribution (from $-\infty$ to ξ)

< *normcdf*(ξ) in MATLAB® >

ξ	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.50000	0.50399	0.50798	0.51197	0.51595	0.51994	0.52392	0.52790	0.53188	0.53586
0.1	0.53983	0.54380	0.54776	0.55172	0.55567	0.55962	0.56356	0.56749	0.57142	0.57535
0.2	0.57926	0.58317	0.58706	0.59095	0.59483	0.59871	0.60257	0.60642	0.61026	0.61409
0.3	0.61791	0.62172	0.62552	0.62930	0.63307	0.63683	0.64058	0.64431	0.64803	0.65173
0.4	0.65542	0.65910	0.66276	0.66640	0.67003	0.67364	0.67724	0.68082	0.68439	0.68793
0.5	0.69146	0.69497	0.69847	0.70194	0.70540	0.70884	0.71226	0.71566	0.71904	0.72240
0.6	0.72575	0.72907	0.73237	0.73565	0.73891	0.74215	0.74537	0.74857	0.75175	0.75490
0.7	0.75804	0.76115	0.76424	0.76730	0.77035	0.77337	0.77637	0.77935	0.78230	0.78524
0.8	0.78814	0.79103	0.79389	0.79673	0.79955	0.80234	0.80511	0.80785	0.81057	0.81327
0.9	0.81594	0.81859	0.82121	0.82381	0.82639	0.82894	0.83147	0.83398	0.83646	0.83891
1.0	0.84134	0.84375	0.84614	0.84849	0.85083	0.85314	0.85543	0.85769	0.85993	0.86214
1.1	0.86433	0.86650	0.86864	0.87076	0.87286	0.87493	0.87698	0.87900	0.88100	0.88298
1.2	0.88493	0.88686	0.88877	0.89065	0.89251	0.89435	0.89617	0.89796	0.89973	0.90147
1.3	0.90320	0.90490	0.90658	0.90824	0.90988	0.91149	0.91308	0.91466	0.91621	0.91774
1.4	0.91924	0.92073	0.92220	0.92364	0.92507	0.92647	0.92785	0.92922	0.93056	0.93189
1.5	0.93319	0.93448	0.93574	0.93699	0.93822	0.93943	0.94062	0.94179	0.94295	0.94408
1.6	0.94520	0.94630	0.94738	0.94845	0.94950	0.95053	0.95154	0.95254	0.95352	0.95449
1.7	0.95543	0.95637	0.95728	0.95818	0.95907	0.95994	0.96080	0.96164	0.96246	0.96327
1.8	0.96407	0.96485	0.96562	0.96638	0.96712	0.96784	0.96856	0.96926	0.96995	0.97062
1.9	0.97128	0.97193	0.97257	0.97320	0.97381	0.97441	0.97500	0.97558	0.97615	0.97670
2.0	0.97725	0.97778	0.97831	0.97882	0.97932	0.97982	0.98030	0.98077	0.98124	0.98169
2.1	0.98214	0.98257	0.98300	0.98341	0.98382	0.98422	0.98461	0.98500	0.98537	0.98574
2.2	0.98610	0.98645	0.98679	0.98713	0.98745	0.98778	0.98809	0.98840	0.98870	0.98899
2.3	0.98928	0.98956	0.98983	0.99010	0.99036	0.99061	0.99086	0.99111	0.99134	0.99158
2.4	0.99180	0.99202	0.99224	0.99245	0.99266	0.99286	0.99305	0.99324	0.99343	0.99361
2.5	0.99379	0.99396	0.99413	0.99430	0.99446	0.99461	0.99477	0.99492	0.99506	0.99520
2.6	0.99534	0.99547	0.99560	0.99573	0.99585	0.99598	0.99609	0.99621	0.99632	0.99643
2.7	0.99653	0.99664	0.99674	0.99683	0.99693	0.99702	0.99711	0.99720	0.99728	0.99736
2.8	0.99744	0.99752	0.99760	0.99767	0.99774	0.99781	0.99788	0.99795	0.99801	0.99807
2.9	0.99813	0.99819	0.99825	0.99831	0.99836	0.99841	0.99846	0.99851	0.99856	0.99861
3.0	0.99865	0.99869	0.99874	0.99878	0.99882	0.99886	0.99889	0.99893	0.99896	0.99900
3.1	0.99903	0.99906	0.99910	0.99913	0.99916	0.99918	0.99921	0.99924	0.99926	0.99929
3.2	0.99931	0.99934	0.99936	0.99938	0.99940	0.99942	0.99944	0.99946	0.99948	0.99950
3.3	0.99952	0.99953	0.99955	0.99957	0.99958	0.99960	0.99961	0.99962	0.99964	0.99965
3.4	0.99966	0.99968	0.99969	0.99970	0.99971	0.99972	0.99973	0.99974	0.99975	0.99976
3.5	0.99977	0.99978	0.99978	0.99979	0.99980	0.99981	0.99981	0.99982	0.99983	0.99983

APPENDIX A.2

Derivation of Hasofer-Lind Reliability Index

From Equation (2.52), we have the first-order Taylor series expansion of $g(\boldsymbol{U})$ at the MPP.

$$\tilde{g}(\boldsymbol{U}) \approx g(\boldsymbol{u}^*) + \sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial u_i}(u_i - u_i^*) \quad \textbf{(1)}$$

From Equation (2.36), we have

$$u_i = \frac{x_i - \mu_{X_i}}{\sigma_{X_i}} \quad \textbf{(2)}$$

The first derivative of Equation (2) is

$$du_i = \frac{dx_i}{\sigma_{X_i}} \quad \textbf{(3)}$$

From Equation (3),

$$\frac{\partial g(\boldsymbol{u}^*)}{\partial u_i} = \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i} \quad \textbf{(4)}$$

Equation (1) can be rewritten as

$$\tilde{g}(\boldsymbol{U}) \approx g(\boldsymbol{u}^*) + \sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i}(u_i - u_i^*) \quad \textbf{(5)}$$

The mean value of $\tilde{g}(\boldsymbol{U})$ is

$$\mu_{\tilde{g}} \approx E[\tilde{g}(\boldsymbol{U})] = E[g(\boldsymbol{u}^*)] + E\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i}u_i\right] - E\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i}u_i^*\right] \quad \textbf{(6)}$$

In Equation (6),

$$E[g(\boldsymbol{u}^*)] = g(\boldsymbol{u}^*) \quad \textbf{(7)}$$

$$E\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i}u_i\right] = E\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}(x_i - \mu_{X_i})\right] \quad \textbf{(8)}$$

$$E\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i}u_i\right] = E\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}x_i\right] - E\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\mu_{X_i}\right] \quad \textbf{(9)}$$

$$E\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i}u_i\right] = E[x_i]\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} - E[\mu_{X_i}]\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \quad \textbf{(10)}$$

$$E\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i}u_i\right] = \mu_{X_i}\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} - \mu_{X_i}\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} = 0 \quad \textbf{(11)}$$

$$E\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i}u_i^*\right] = \sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\sigma_{X_i}u_i^* \quad \textbf{(12)}$$

Using Equations (7), (11) and (12) in Equation (6) we obtain,

$$\mu_{\tilde{g}} \approx E[\tilde{g}(\boldsymbol{U})] = g(\boldsymbol{u}^*) - \sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \sigma_{X_i} u_i^* \quad \textbf{(13)}$$

The standard deviation of $\tilde{g}(\boldsymbol{U})$ is

$$\sigma_{\tilde{g}} = \sqrt{Var[\tilde{g}(\boldsymbol{U})]} \quad \textbf{(14)}$$

In Equation (14) using Equation (5),

$$Var[\tilde{g}(\boldsymbol{U})] = Var[g(\boldsymbol{u}^*)] + Var\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \sigma_{X_i}(u_i - u_i^*)\right] \quad \textbf{(15)}$$

$$Var[\tilde{g}(\boldsymbol{U})] = Var[g(\boldsymbol{u}^*)] + Var\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \sigma_{X_i} u_i\right] - Var\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \sigma_{X_i} u_i^*\right] \quad \textbf{(16)}$$

In Equation (16),

$$Var[g(\boldsymbol{u}^*)] = 0 \quad \textbf{(17)}$$

$$Var\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \sigma_{X_i} u_i\right] = Var\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} (x_i - \mu_{X_i})\right] \quad \textbf{(18)}$$

$$Var\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \sigma_{X_i} u_i\right] = Var\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} x_i\right] - Var\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \mu_{X_i}\right] \quad \textbf{(19)}$$

$$Var\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \sigma_{X_i} u_i\right] = \sum_{i=1}^{n} \left(\frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\right)^2 Var[x_i] = \sum_{i=1}^{n} \left(\frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\right)^2 \sigma_{X_i}^2 \quad \textbf{(20)}$$

$$Var\left[\sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \sigma_{X_i} u_i^*\right] = 0 \quad \textbf{(21)}$$

Using Equations (17), (20) and (21) in Equation (16),

$$Var[\tilde{g}(\boldsymbol{U})] = \sum_{i=1}^{n} \left(\frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\right)^2 \sigma_{X_i}^2 \quad \textbf{(22)}$$

Using Equation (22) in Equation (14),

$$\sigma_{\tilde{g}} = \sqrt{Var[\tilde{g}(\boldsymbol{U})]} = \sqrt{\sum_{i=1}^{n} \left(\frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\right)^2 \sigma_{X_i}^2} \quad \textbf{(23)}$$

Using Equations (13) and (23), the Hasofer-Lind reliability index is obtained as

$$\beta_s = \frac{\mu_{\tilde{g}}}{\sigma_{\tilde{g}}} = \frac{g(\boldsymbol{u}^*) - \sum_{i=1}^{n} \frac{\partial g(\boldsymbol{u}^*)}{\partial x_i} \sigma_{X_i} u_i^*}{\sqrt{\sum_{i=1}^{n} \left(\frac{\partial g(\boldsymbol{u}^*)}{\partial x_i}\right)^2 \sigma_{X_i}^2}} \quad \textbf{(24)}$$

APPENDIX A.3

Deterministic Design Optimization of a Cantilever Beam Matlab Code

main.m

```
% Deterministic Design Optimization of a Cantilever Beam
% Filename: main.m
% Date     : 09.10.2010
% Code was written by MUHAMMET NASIF KURU
% Note     : objective.m and nonlcon.m must be in the same directory
% Output : Area = w*t, width and thickness values
clc; clear;
% Optimization Variables : w, t
x0 = [4.0 ; 4.0];
options = optimset('Largescale','off','Display','iter');
[x, fval] = fmincon(@objective, x0, [], [], [], [], [], [], @nonlcon,options)
```

objective.m

```
function f = objective(x)
  f = x(1) * x(2); % w * t
```

nonlcon.m

```
function [c, ceq] = nonlcon(x)
  % Optimization Variables : w, t
  w = x(1);
  t = x(2);
  % The random variables (R, X, Y) are fixed at their means.
  R = 40000;
  X = 500;
  Y = 1000;
  % Maximum Normal Stress
  S = 600 * Y / (w * t^2) + 600 * X / (w^2 * t);
  % Limit State Function
  safety_factor = 1.5;
```

```
g = R - safety_factor * S;
% Constraints
c(1) = -g;
ceq = [];
```

APPENDIX A.4

Reliability Based Design Optimization of a Cantilever Beam Matlab Code (Analytical RIA)

main.m

```
% Reliability Based Design Optimization of a Cantilever Beam
% Reliability Index Approach and First Order Reliability Method (FORM) is
% used.
% Filename : main.m
% Date       : 04.04.2011
% Code was written by MUHAMMET NASIF KURU
% Output : Area = w*t, width and thickness values
clc; clear;
% Optimization Variables : w, t
d0 = [4.0; 4.0];
options = optimset('Display', 'iter','MaxFunEvals',2000,'TolCon',1e-6,'TolFun',1e-
6,'TolX',1e-6);
[d, fval] = fmincon(@objective, d0, [], [], [], [], [], [], @nonlcon, options)
```

objective.m

```
function f = objective(d)
  f = d(1) * d(2);
```

stress.m

```
function f = stress(a)
  R = a(1);
  X = a(2);
  Y = a(3);
  w = a(4);
  t  = a(5);
  f  = R-(600*Y/(w*(t^2))+600*X/((w^2)*t));
```

displacement.m

```
function f = displacement(a)
  E = a(1);
  X = a(2);
  Y = a(3);
  w = a(4);
  t = a(5);
  D0 = 2.2535;
  L = 100.0;
  D = 4 * L^3 * sqrt((Y/t^2)^2+(X/w^2)^2) / (E * w * t)
  f = D0 - D;
```

beta_stress.m

```
function beta_s = beta_stress(d)
% This function evaluates the beta stress value for the given design and
% returns the beta_s value
% R, X, Y
x_mean     = [40000,500,1000];          % Mean values
x_standard = [2000, 100, 100];          % Standard deviations
h = 0.1;                                % For gradient calculation increment amount
convergence = 10;                       % To start the while loop, it is necessary
x = [x_mean(1), x_mean(2), x_mean(3)];  % Initial design point
u = [0, 0, 0];                          % Initial design point in the standard normal space
[row, col] = size(x);
i = 1;
while (convergence >= 0.001)
  display('Iteration: ');
  a = [x,d'];                           % R X Y w t
  m_g = stress(a);                      % Limit state function's value at design point
  % Gradient calculation using central finite differences method
  for k = 1:col
    temp = x(k);
    x(k) = (u(k) + h) * x_standard(k) + x_mean(k);
    P2 = stress([x,d']);
```

```
    x(k) = (u(k) - h) * x_standard(k) + x_mean(k);
    P1 = stress([x,d']);
    grad(k) = ((P2 - P1) / (2 * h)) / x_standard(k);
    x(k) = temp;
  end
  % End gradient calculation
  sigma_g = sqrt(sum((grad .* x_standard).^2));
  if i == 1
     display('MVFOSM Beta:');
     beta(i) = m_g / sigma_g
     alfa = - (grad .* x_standard) ./ sigma_g;
  else
     display('FORM Beta :')
     beta_upper = sum(grad .* x_standard .* u);
     beta(i) = (m_g - beta_upper) / sigma_g
     alfa = - (grad .* x_standard) ./ sigma_g;
  end
  x = x_mean + beta(i) .* x_standard .* alfa;      % Compute a new design point
  u = (x - x_mean) ./ x_standard;
  % Check beta convergence
  if i == 1
     convergence = 10;
  else
     convergence = abs(beta(i) - beta(i-1)) / beta(i-1);
  end
  i = i + 1;
end
mpp   = [x(1), x(2), x(3)];              % Most probable failure point (MPP)
a = [mpp,d'];
mpp_g = stress(a);                        % Limit state function's value at the MPP
beta_s = beta(size(beta,2));              % Shortest distance to the MPP
```

beta_displacement.m

```
function beta_d = beta_displacement(d)
```

```
% This function evaluates the beta displacement value for the given design and
% returns the beta_d value
% E, X, Y
x_mean     = [29E6,500,1000];          % Mean values
x_standard = [1.45E6, 100, 100];       % Standard deviations
h = 0.1;                               % For gradient calculation increment amount
convergence = 10;                      % To start the while loop, it is necessary
x = [x_mean(1), x_mean(2), x_mean(3)];   % Initial design point
u = [0, 0, 0];                         % Initial design point in the standard normal space
[row, col] = size(x);
i = 1;
while (convergence >= 0.001)
  display('Iteration: ');
  a = [x,d'];                          % E X Y w t
  m_g = displacement(a);    % Limit state function's value at design point
  % Gradient calculation using central finite differences method
  for k = 1:col
    temp = x(k);
    x(k) = (u(k) + h) * x_standard(k) + x_mean(k);
    P2 = displacement([x,d']);
    x(k) = (u(k) - h) * x_standard(k) + x_mean(k);
    P1 = displacement([x,d']);
    grad(k) = ((P2 - P1) / (2 * h)) / x_standard(k);
    x(k) = temp;
  end
  % End gradient calculation
  sigma_g = sqrt(sum((grad .* x_standard).^2));
  if i == 1
     display('MVFOSM Beta:');
     beta(i) = m_g / sigma_g
     alfa = - (grad .* x_standard) ./ sigma_g;
  else
     display('FORM Beta :')
     beta_upper = sum(grad .* x_standard .* u);
```

```
    beta(i) = (m_g - beta_upper) / sigma_g
    alfa = - (grad .* x_standard) ./ sigma_g;
  end
  x = x_mean + beta(i) .* x_standard .* alfa;      % Compute a new design point
  u = (x - x_mean) ./ x_standard;
  % Check beta convergence
  if i == 1
    convergence = 10;
  else
    convergence = abs(beta(i) - beta(i-1)) / beta(i-1);
  end
  i = i + 1;
end
mpp   = [x(1), x(2), x(3)];          % Most probable failure point (MPP)
a = [mpp,d'];
mpp_g = displacement(a);             % Limit state function's value at the MPP
beta_d = beta(size(beta,2));         % Shortest distance to the MPP
```

nonlcon.m

```
function [c, ceq] = nonlcon(d)
beta_t = 3.0;
beta_s = beta_stress(d)
beta_d = beta_displacement(d)
c(1) = beta_t - beta_s;
c(2) = beta_t - beta_d;
ceq = [];
```

APPENDIX A.5

Reliability Based Design Optimization of a Cantilever Beam Matlab Code (Computational RIA)

nonlcon.m

```
function nonlcon()
% This function evaluates the beta value for the given design
% Output : output.dat
display('Beta Stress Calculation')
display('Optimization Variables')
d= [w, t];
w = d(1)
t = d(2)
% R, X, Y
x_mean     = [40000,500,1000];          % Mean values
x_standard = [2000, 100, 100];          % Standard deviations
h = 0.1;                                % For gradient calculation increment amount
convergence = 10;                       % To start the while loop, it is necessary
x = [x_mean(1), x_mean(2), x_mean(3)];  % Initial design point
u = [0, 0, 0];                          % Initial design point in the standard normal space
[row, col] = size(x);
i = 1;
while (convergence >= 0.01)
   display('Iteration : ');
   grad = [];
   vonmises = vonmises_calculate([x,[w, t]])    % R X Y w t
   m_g = x(1) - vonmises                % Limit state function's value at design point
   % Gradient calculation using central finite differences method
   for k = 1:col
     temp = x(k);
     x(k) = (u(k) + h) * x_standard(k) + x_mean(k)
     if (k ~= 1)
       vonmises = vonmises_calculate([x,[w,t]])
```

```
  end
  P2 = x(1) - vonmises
  x(k) = (u(k) - h) * x_standard(k) + x_mean(k)
  if (k ~= 1)
    vonmises = vonmises_calculate([x,[w,t]])
  end
  P1 = x(1) - vonmises
  grad(k) = ((P2 - P1) / (2 * h)) / x_standard(k)
  x(k) = temp;
end
% End gradient calculation
display('Gradient')
grad
sigma_g = sqrt(sum((grad .* x_standard).^2));
if i == 1
   display('MVFOSM Beta :')
   beta(i) = m_g / sigma_g
   alfa = - (grad .* x_standard) ./ sigma_g
else
   display('FORM Beta :')
   beta_upper = sum(grad .* x_standard .* u);
   beta(i) = (m_g - beta_upper) / sigma_g
   alfa = - (grad .* x_standard) ./ sigma_g
end
display('New Design Point')
x = x_mean + beta(i) .* x_standard .* alfa          % Compute a new design point.
u = (x - x_mean) ./ x_standard
% Check beta convergence
if i == 1
   convergence = 10;
else
   convergence = abs(beta(i) - beta(i-1)) / beta(i-1)
end
i = i + 1;
```

```
end
mpp   = [x(1), x(2), x(3)]                    % Most probable failure point (MPP)
vonmises = vonmises_calculate([mpp,[w,t]])
mpp_g = mpp(1) - vonmises                     % Limit state function's value at the MPP
converged_beta = beta(size(beta,2))           % Shortest distance to the MPP
f_write = fopen('output.dat', 'w');
% converged_beta, R, S, g = R - S, X, Y
fprintf(f_write, '%f\n%f\n%f\n%f\n%f\n%f\n', converged_beta, mpp(1), vonmises,
mpp_g, mpp(2), mpp(3));
fclose(f_write);
exit
```

vonmises_calculate.m

```
function vonmises = vonmises_calculate(a)
  E = 29 * 10^6;
  % Write the values that must be updated for the given design to "mtlb.txt"
  fid = fopen('mtlb.txt', 'w');
  % w, t , E, x, y
  fprintf(fid, '%f %f %f %f %f', a(4), a(5), E, a(2), a(3));
  fclose(fid);
  % Updates the abaqus.py for the given input file "mtlb.txt"
  !python writeinput_beam.py
  % Call abaqus to calculate the maximum normal stress value
  !abq671.bat cae noGUI=abaqus.py
  % Takes the maximum normal stress value from the abaqus output file
  % "beam_stress.rpt"
  !python getMises_beam.py
  % Assigns the maximum normal stress value to the variable "vonmises"
  fid = fopen('mises.dat', 'r');
  vonmises = fscanf(fid, '%f', 1);
  fclose(fid);
```

writeinput_beam.py

```
import re, os
```

```
a = open("abaqus.py", "r")
b = open("abaqus_temp.py", "w")
matlab = open("mtlb.txt","r")
e = matlab.read()
value = e.split(" ")
k = 0
for i in value:
    value[k] = float(i)
    k = k + 1
for i in a.readlines():
    w = re.search("(.*Rect-Profile\'\,)(.*)\)",i)
    E=re.search("(mdb.models\[\'Model\1\'\]\.materials\[\'Aliminium\'\]\.Elastic\(
table=\(\()(.*)\,", i)
    f1=re.search("(region=region, cf3=)(.*)\,(.*)", i)  # Load x
    f2=re.search("(region=region, cf2=-)(.*)\,(.*)", i) # Load y
    if w:
        w_write = w.group(1) + " a=" + str(value[0]) + ", b=" + str(value[1]) + ")\n"
        b.write(w_write)
    elif E:
        E_write = E.group(1) + str(value[2]) + ", 0.29), \n"
        b.write(E_write)
    elif f1:
        f1_write = f1.group(1) + str(value[3]) + "," + f1.group(3) + "\n"
        b.write(f1_write)
    elif f2:
        f2_write = f2.group(1) + str(value[4]) + "," + f2.group(3) + "\n"
        b.write(f2_write)
    else:
        b.write(i)
a.close()
b.close()
os.remove("abaqus.py")
os.rename("abaqus_temp.py", "abaqus.py")
```

getMises_beam.py

```
# Input  : beam_stress.rpt
# Output : mises.dat
import re, os
a = open("beam_stress.rpt", "r")
b = open("mises.dat", "w")
for i in a.readlines():
    mises = re.search(".*Maximum(.*)", i)
    if mises:
        liste = mises.group(1)
        liste1 = liste.strip()
        liste2 = liste1.split(" ")
        v_mises = float(liste2[0])
        b.write(str(v_mises))
a.close()
b.close()
os.remove("beam_stress.rpt")
```

abaqus.py

```
# -*- coding: mbcs -*-
# Abaqus/CAE Version 6.7-1 replay file
# Internal Version: 2007_05_01-12.35.33 79448
# Run by Owner on Sun Feb 27 09:49:11 2011
# from driverUtils import executeOnCaeGraphicsStartup
# executeOnCaeGraphicsStartup()
#: Executing "onCaeGraphicsStartup()" in the site directory ...
from abaqus import *
from abaqusConstants import *
session.Viewport(name='Viewport: 1', origin=(0.0, 0.0), width=133.858588412404,
    height=162.841807678342)
session.viewports['Viewport: 1'].makeCurrent()
session.viewports['Viewport: 1'].maximize()
from caeModules import *
from driverUtils import executeOnCaeStartup
```

```
executeOnCaeStartup()
Mdb()
#: A new model database has been created.
#: The model "Model-1" has been created.
session.viewports['Viewport: 1'].setValues(displayedObject=None)
session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=OFF,
    engineeringFeatures=OFF)
s = mdb.models['Model-1'].ConstrainedSketch(name='__profile__',
    sheetSize=300.0)
g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
s.setPrimaryObject(option=STANDALONE)
s.Line(point1=(0.0, 0.0), point2=(100.0, 0.0))
s.HorizontalConstraint(entity=g[2])
p = mdb.models['Model-1'].Part(name='Part-1', dimensionality=THREE_D,
    type=DEFORMABLE_BODY)
p = mdb.models['Model-1'].parts['Part-1']
p.BaseWire(sketch=s)
s.unsetPrimaryObject()
p = mdb.models['Model-1'].parts['Part-1']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
del mdb.models['Model-1'].sketches['__profile__']
session.viewports['Viewport: 1'].partDisplay.setValues(sectionAssignments=ON,
    engineeringFeatures=ON)
mdb.models['Model-1'].Material(name='Aliminium')
mdb.models['Model-1'].materials['Aliminium'].Elastic(table=((29000000.0, 0.29),
    ))
mdb.models['Model-1'].RectangularProfile(name='Rect-Profile', a=2.4, b=3.9673)
mdb.models['Model-1'].BeamSection(name='BeamSection', profile='Rect-Profile',
    integration=DURING_ANALYSIS, poissonRatio=0.0, material='Aliminium',
    temperatureVar=LINEAR)
p = mdb.models['Model-1'].parts['Part-1']
e = p.edges
edges = e.getSequenceFromMask(mask=('[#1 ]', ), )
region = regionToolset.Region(edges=edges)
```

```
p = mdb.models['Model-1'].parts['Part-1']
p.SectionAssignment(region=region, sectionName='BeamSection', offset=0.0)
a = mdb.models['Model-1'].rootAssembly
session.viewports['Viewport: 1'].setValues(displayedObject=a)
a1 = mdb.models['Model-1'].rootAssembly
a1.DatumCsysByDefault(CARTESIAN)
p = mdb.models['Model-1'].parts['Part-1']
a1.Instance(name='Part-1-1', part=p, dependent=OFF)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(
    adaptiveMeshConstraints=ON)
mdb.models['Model-1'].StaticStep(name='Step-1', previous='Initial',
    description='Beam-Step', initialInc=0.1)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(step='Step-1')
session.viewports['Viewport: 1'].assemblyDisplay.setValues(loads=ON, bcs=ON,
    predefinedFields=ON, connectors=ON, adaptiveMeshConstraints=OFF)
a = mdb.models['Model-1'].rootAssembly
v1 = a.instances['Part-1-1'].vertices
verts1 = v1.getSequenceFromMask(mask=('[#1 ]', ), )
region = regionToolset.Region(vertices=verts1)
mdb.models['Model-1'].DisplacementBC(name='BC-1', createStepName='Step-1',
    region=region, u1=0.0, u2=0.0, u3=0.0, ur1=0.0, ur2=0.0, ur3=0.0,
    amplitude=UNSET, fixed=OFF, distributionType=UNIFORM, fieldName='',
    localCsys=None)
session.viewports['Viewport: 1'].view.setValues(nearPlane=143.167,
    farPlane=256.833, width=75.1627, height=63.0594, cameraPosition=(203.29,
    29.8088, 124.954), cameraUpVector=(-0.145208, 0.987729, -0.0574938))
a = mdb.models['Model-1'].rootAssembly
v1 = a.instances['Part-1-1'].vertices
verts1 = v1.getSequenceFromMask(mask=('[#2 ]', ), )
region = regionToolset.Region(vertices=verts1)
mdb.models['Model-1'].ConcentratedForce(name='Load-x',    createStepName='Step-
1',
    region=region, cf3=715.061054, localCsys=None)
a = mdb.models['Model-1'].rootAssembly
```

```
v1 = a.instances['Part-1-1'].vertices
verts1 = v1.getSequenceFromMask(mask=('[#2 ]', ), )
region = regionToolset.Region(vertices=verts1)
mdb.models['Model-1'].ConcentratedForce(name='Load-y',    createStepName='Step-
1',
   region=region, cf2=-1130.117876, localCsys=None)
session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=ON, loads=OFF,
   bcs=OFF, predefinedFields=OFF, connectors=OFF)
session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(
   meshTechnique=ON)
a = mdb.models['Model-1'].rootAssembly
partInstances =(a.instances['Part-1-1'], )
a.seedPartInstance(regions=partInstances, size=0.01, deviationFactor=0.1)
elemType1 = mesh.ElemType(elemCode=B31, elemLibrary=STANDARD)
a = mdb.models['Model-1'].rootAssembly
e1 = a.instances['Part-1-1'].edges
edges1 = e1.getSequenceFromMask(mask=('[#1 ]', ), )
pickedRegions =(edges1, )
a.setElementType(regions=pickedRegions, elemTypes=(elemType1, ))
elemType1 = mesh.ElemType(elemCode=B31, elemLibrary=STANDARD)
a = mdb.models['Model-1'].rootAssembly
e1 = a.instances['Part-1-1'].edges
edges1 = e1.getSequenceFromMask(mask=('[#1 ]', ), )
pickedRegions =(edges1, )
a.setElementType(regions=pickedRegions, elemTypes=(elemType1, ))
a = mdb.models['Model-1'].rootAssembly
partInstances =(a.instances['Part-1-1'], )
a.generateMesh(regions=partInstances)
p = mdb.models['Model-1'].parts['Part-1']
session.viewports['Viewport: 1'].setValues(displayedObject=p)
p = mdb.models['Model-1'].parts['Part-1']
e = p.edges
edges = e.getSequenceFromMask(mask=('[#1 ]', ), )
region=regionToolset.Region(edges=edges)
```

```
p = mdb.models['Model-1'].parts['Part-1']
p.assignBeamSectionOrientation(region=region,    method=N1_COSINES,    n1=(0.0,
0.0, -1.0))
#: Beam orientations have been assigned to the selected regions.
a = mdb.models['Model-1'].rootAssembly
session.viewports['Viewport: 1'].setValues(displayedObject=a)
a1 = mdb.models['Model-1'].rootAssembly
a1.regenerate()
session.viewports['Viewport: 1'].assemblyDisplay.setValues(mesh=OFF)
session.viewports['Viewport: 1'].assemblyDisplay.meshOptions.setValues(
    meshTechnique=OFF)
mdb.Job(name='Job-Beam', model='Model-1', type=ANALYSIS,
    explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE,
    description='Beam job', parallelizationMethodExplicit=DOMAIN,
    multiprocessingMode=DEFAULT, numDomains=1, userSubroutine='',
numCpus=1,
    preMemory=256.0,standardMemory=256.0,standardMemoryPolicy=
MODERATE,
    scratch='', echoPrint=OFF, modelPrint=OFF, contactPrint=OFF,
    historyPrint=OFF)
mdb.jobs['Job-Beam'].submit(consistencyChecking=OFF)
#: The job input file "Job-Beam.inp" has been submitted for analysis.
#: Job Job-Beam: Analysis Input File Processor completed successfully.
#: Job Job-Beam: Abaqus/Standard completed successfully.
#: Job Job-Beam completed successfully.
o3 = session.openOdb(name='Job-Beam.odb')
#: Model: Job-Beam.odb
#: Number of Assemblies:              1
#: Number of Assembly instances: 0
#: Number of Part instances:          1
#: Number of Meshes:                  1
#: Number of Element Sets:           1
#: Number of Node Sets:               1
#: Number of Steps:                   1
```

```
session.viewports['Viewport: 1'].setValues(displayedObject=o3)
odb = session.odbs['Job-Beam.odb']
session.fieldReportOptions.setValues(printXYData=OFF, printTotal=OFF)
session.writeFieldReport(fileName='beam_stress.rpt', append=OFF,
    sortItem='S.Mises', odb=odb, step=0, frame=6,
    outputPosition=INTEGRATION_POINT,variable=(('S', INTEGRATION_POINT,
((
    INVARIANT, 'Mises'), )), ))
```

APPENDIX A.6

Reliability Based Design Optimization of a Cantilever Beam Matlab Code (Analytical PMA)

main.m

```
% Reliability Based Design Optimization of a Cantilever Beam
% Performance Measure Approach and Advanced Mean Value (AMV) Method is
% used.
% Filename : main.m
% Date       : 28.10.2010
% Code was written by MUHAMMET NASIF KURU
% Note    : objective.m, performance.m and nonlcon.m must be in the same directory
% Output : Area = w*t, width and thickness values
clc;clear;
% Optimization Variables : w, t
d0 = [4.0; 4.0];
options = optimset('Display', 'iter','MaxFunEvals',2000,'TolCon',1e-6,'TolFun',1e-
6,'TolX',1e-6);
[d, fval] = fmincon(@objective, d0, [], [], [], [], [], [], @nonlcon, options)
```

objective.m

```
function f = objective(d)
  f = d(1) * d(2);
```

performance.m

```
function f = performance(a)
  R = a(1);
  X = a(2);
  Y = a(3);
  w = a(4);
  t = a(5);
  f = R-(600*Y/(w*(t^2))+600*X/((w^2)*t));
```

nonlcon.m

```
function [c, ceq] = nonlcon(d)
% This function evaluates the g value for the given design
% R, X, Y
x_mean     = [40000,500,1000];        % Mean values
x_standard = [2000, 100, 100];        % Standard deviations
beta_t = 3.0;                         % Target reliability index
h = 0.1;                              % For gradient calculation increment amount
convergence = 10;                      % To start the while loop, it is necessary
x = [x_mean(1), x_mean(2), x_mean(3)];     % Initial design point
u = [0, 0, 0];     % Initial design point in the standard normal space
[row, col] = size(x);
i = 1;
while (convergence >= 10e-3)
  % Gradient calculation using central finite differences method
  for k = 1:col
    temp = x(k);
    x(k) = (u(k) + h) * x_standard(k) + x_mean(k);
    P2 = performance([x,d']);
    x(k) = (u(k) - h) * x_standard(k) + x_mean(k);
    P1 = performance([x,d']);
    grad(k) = ((P2 - P1) / (2 * h));   % Gradient in U-space
    x(k) = temp;
  end
  display('Gradient')
  grad_x = grad ./ x_standard        % Gradient in X-space
  % End gradient calculation
  norm = sqrt(sum(grad.^2));
  n = - grad / norm                  % Normalized steepest descent direction
  u = beta_t * n
  x = u .* x_standard + x_mean    % New Design Point
  g(i) = performance([x,d'])          % Performance Function's Value
  if i == 1
    convergence = 10;
```

```
  else
     convergence = abs(g(i) - g(i-1));
  end
  i = i + 1;
end
mpp = x            % Most probable failure point
c(1) = -g(i-1);      % Negative value of performance function's value at the MPP
ceq = [];
```

APPENDIX A.7

Reliability Based Design Optimization of a Cantilever Beam Matlab Code (Computational PMA)

nonlcon.m

```
function nonlcon()
% This function evaluates the g value for the given design
% Output : output.dat
display('Performance Function Calculation')
display('Optimization Variables')
d = [w, t];
w = d(1)
t = d(2)
% R, X, Y
x_mean     = [40000,500,1000];          % Mean values
x_standard = [2000, 100, 100];          % Standard deviations
h = 0.1;                                % For gradient calculation increment amount
beta_t = 3.0;                           % Target reliability index
h = 0.1;                                % For gradient calculation increment amount
convergence = 10;                       % To start the while loop, it is necessary
x = [x_mean(1), x_mean(2), x_mean(3)];   % Initial design point
u = [0, 0, 0];                          % Initial design point in the standard normal space
[row, col] = size(x);
i = 1;
while (convergence >= 0.2)
  display('Iteration : ');
  grad = [];
  % Gradient calculation using central finite differences method
  vonmises = vonmises_calculate([x,[w, t]])    % R X Y w t
  for k = 1:col
    temp = x(k);
    x(k) = (u(k) + h) * x_standard(k) + x_mean(k)
    if (k ~= 1)
```

```
      vonmises = vonmises_calculate([x,[w,t]])
    end
    P2 = x(1) - vonmises
    x(k) = (u(k) - h) * x_standard(k) + x_mean(k)
    if (k ~= 1)
      vonmises = vonmises_calculate([x,[w,t]])
    end
    P1 = x(1) - vonmises
    grad(k) = ((P2 - P1) / (2 * h)) / x_standard(k)          % Gradient in X-space
    x(k) = temp;
  end
  % End gradient calculation
  display('Gradient in X-Space')
  grad
  display('Gradient in U-Space')
  grad_u = grad .* x_standard
  norm = sqrt(sum(grad_u.^2))
  n = - grad_u / norm                    % Normalized steepest descent direction
  u = beta_t * n
  display('New Design Point')
  x = u .* x_standard + x_mean          % New design point
  vonmises = vonmises_calculate([x,[w,t]])
  g(i) = x(1) - vonmises                 % Performance function's value
  % Check g convergence
  if i == 1
    convergence = 10;
  else
    convergence = abs(g(i) - g(i-1))
  end
  i = i + 1;
end
mpp = x                    % Most probable failure point (MPP)
converged_g = g(i-1)       % Performance function's value at the MPP
f_write = fopen('output.dat', 'w');
```

```
% converged_g, R, S, X, Y
fprintf(f_write, '%f\n%f\n%f\n%f\n%f\n', converged_g, mpp(1), vonmises, mpp(2),
mpp(3));
fclose(f_write);
%
exit
```

vonmises_calculate.m : Appendix A.5

writeinput_beam.py : Appendix A.5

getMises_beam.py : Appendix A.5

abaqus.py : Appendix A.5

APPENDIX A.8

Reliability Based Design Optimization of a Generic Aircraft Wing Matlab Code

nonlcon.m

```
function nonlcon()
% Output : output.dat
beta_stress();
beta_displacement();
exit
```

beta_stress.m

```
function beta_stress()
% This function evaluates the beta value for the given design
display('Beta Stress Calculation')
% R
x_mean     = [400];        % Mean value
x_standard = [20];         % Standard deviation
h = 0.1;                   % For gradient calculation increment amount
convergence = 10;          % To start the while loop, it is necessary
x = [x_mean(1)];           % Initial design point
u = [0];                   % Initial design point in the standard normal space
[row, col] = size(x);
i = 1;
while (convergence > 0.1)
   display('Iteration : ');
   grad = [];
   vonmises = vonmises_calculate(x)       % R
   m_g = x(1) - vonmises              % Limit state function's value at design point
   % Gradient calculation using central finite differences method
   for k = 1:col
     temp = x(k);
     x(k) = (u(k) + h) * x_standard(k) + x_mean(k)
     if (k ~= 1)
```

```
    vonmises = vonmises_calculate(x)
  end
  P2 = x(1) - vonmises
  x(k) = (u(k) - h) * x_standard(k) + x_mean(k)
  if (k ~= 1)
    vonmises = vonmises_calculate(x)
  end
  P1 = x(1) - vonmises
  grad(k) = ((P2 - P1) / (2 * h)) / x_standard(k)
  x(k) = temp;
end
% End gradient calculation
display('Gradient')
grad
sigma_g = sqrt(sum((grad .* x_standard).^2));
if i == 1
   display('MVFOSM Beta :')
   beta(i) = m_g / sigma_g
   alfa = - (grad .* x_standard) ./ sigma_g
else
   display('FORM Beta :')
   beta_upper = sum(grad .* x_standard .* u);
   beta(i) = (m_g - beta_upper) / sigma_g
   alfa = - (grad .* x_standard) ./ sigma_g
end
display('New Design Point')
x = x_mean + beta(i) .* x_standard .* alfa       % Compute a new design point.
u = (x - x_mean) ./ x_standard
% Check beta convergence
if i == 1
   convergence = 10;
else
   convergence = abs(beta(i) - beta(i-1)) / beta(i-1)
end
```

```
   i = i + 1;
end
mpp   = [x(1)]                          % Most probable failure point (MPP)
mpp_g = mpp(1) - vonmises               % Limit state function's value at the MPP
converged_beta = beta(size(beta,2))     % Shortest distance to the MPP
f_write = fopen('output.dat', 'w');
% converged_beta, R, S, g = R - S
fprintf(f_write, '%f\n%f\n%f\n%f\n%f\n%f\n', converged_beta, mpp(1), vonmises,
mpp_g);
fclose(f_write);
```

vonmises_calculate.m

```
function vonmises = vonmises_calculate(a)
  E = 70000;
  % Write the values that must be updated for the given design to "mtlb.txt"
  fid = fopen('mtlb.txt', 'w');
  % E
  fprintf(fid, '%f', E);
  fclose(fid);
  % Updates the espana.py for the given input file "mtlb.txt"
  !python writeinput_wing.py
  % Call abaqus to calculate the maximum von mises stress value
  !abq671.bat cae noGUI=espana.py
  % Takes the mass and frequency values from the file "espana_statik.dat"
  % and writes them into the file "mass_freq.dat"
  !python mass_freq.py
  % Takes the maximum von mises stress value from the abaqus output file
  % "espana_mises.rpt"
  !python getMises_wing.py
  % Assigns the maximum von Mises stress value to the variable "vonmises"
  fid = fopen('mises.dat', 'r');
  vonmises = fscanf(fid, '%f', 1);
  fclose(fid);
```

writeinput_wing.py

```
import re, os
a = open("espana.py", "r")
b = open("espana_temp.py", "w")
matlab = open("mtlb.txt","r")
e = float(matlab.read())
for i in a.readlines():
   E=re.search("(mdb.models\[\'Model\-1\'\]\.materials\[\'Material\-
1\'\]\.Elastic\(table=\(\()(.*)\,", i)
   if E:
      E_yazdir = E.group(1) + str(e) + ", 0.33),\n"
      b.write(E_yazdir)
   else:
      b.write(i)
a.close()
b.close()
os.remove("espana.py")
os.rename("espana_temp.py", "espana.py")
```

mass_freq.py

```
import re
a = open("espana_statik.dat", "r")
f = open("mass_freq.dat", "w")
list_a = a.readlines()
for i in list_a:
   search_mass = re.search("TOTAL MASS OF MODEL", i)
   search_freq = re.search(" MODE NO      EIGENVALUE", i)
   if search_mass:
      b = list_a.index(i)
      mass = float(list_a[b + 2])
   elif search_freq:
      c = list_a.index(i)
      freq_list = list_a[c + 4].split()
      freq = float(freq_list[3])
```

```
f.write(str(mass) + "\n" + str(freq))
f.close()
a.close()
```

getMises_wing.py

```
# Input  : espana_mises.rpt
# Output : mises.dat
import re, os
a = open("espana_mises.rpt", "r")
b = open("mises.dat", "w")
for i in a.readlines():
    mises = re.search(".*Maximum(.*)", i)
    if mises:
        stress = float(mises.group(1))
        b.write(str(stress))
        break
a.close()
b.close()
os.remove("espana_mises.rpt")
```

beta_displacement.m

```
function beta_displacement()
% This function evaluates the beta value for the given design
display('Beta Displacement Calculation')
D0 = 187;
% E
x_mean      = [70000];       % Mean value
x_standard = [350];          % Standard deviation
h = 0.1;                     % For gradient calculation increment amount
convergence = 10;            % To start the while loop, it is necessary
x = [x_mean(1)];             % Initial design point
u = [0];                     % Initial design point in the standard normal space
[row, col] = size(x);
i = 1;
```

```
while (convergence > 0.1)
    display('Iteration : ');
    grad = [];
    disp = displacement(x)              % E
    m_g = D0 - disp                     % Limit state function's value at design point
    % Gradient calculation using central finite differences method
    for k = 1:col
      temp = x(k);
      x(k) = (u(k) + h) * x_standard(k) + x_mean(k)
      disp = displacement(x)
      P2 = D0 - disp
      x(k) = (u(k) - h) * x_standard(k) + x_mean(k)
      disp = displacement(x)
      P1 = D0 - disp
      grad(k) = ((P2 - P1) / (2 * h)) / x_standard(k)
      x(k) = temp;
    end
    % End gradient calculation
    display('Gradient')
    grad
    sigma_g = sqrt(sum((grad .* x_standard).^2));
    if i == 1
       display('MVFOSM Beta :')
       beta(i) = m_g / sigma_g
       alfa = - (grad .* x_standard) ./ sigma_g
    else
       display('FORM Beta :')
       beta_upper = sum(grad .* x_standard .* u);
       beta(i) = (m_g - beta_upper) / sigma_g
       alfa = - (grad .* x_standard) ./ sigma_g
    end
    display('New Design Point')
    x = x_mean + beta(i) .* x_standard .* alfa          % Compute a new design point.
    u = (x - x_mean) ./ x_standard
```

```
    % Check beta convergence
    if i == 1
        convergence = 10;
    else
        convergence = abs(beta(i) - beta(i-1)) / beta(i-1)
    end
    i = i + 1;
end
mpp   = [x(1)]                          % Most probable failure point (MPP)
disp = displacement(mpp)
mpp_g = D0 - disp                       % Limit state function's value at the MPP
converged_beta = beta(size(beta,2))   % Shortest distance to the MPP
f_write = fopen('output.dat', 'a');
% converged_beta, displacement, mpp_g, E
fprintf(f_write,  '%f\n%f\n%f\n%f\n%f\n%f\n',  converged_beta,  disp,  mpp_g,
mpp(1));
fclose(f_write);
```

displacement.m

```
function disp = displacement(x)
    % Write the values that must be updated for the given design to "mtlb.txt"
    fid = fopen('mtlb.txt', 'w');
    % E
    fprintf(fid, '%f', x(1));
    fclose(fid);
    % Updates the espana.py for the given input file "mtlb.txt"
    !python writeinput_wing.py
    % Call abaqus to calculate the maximum displacement value
    !abq671.bat cae noGUI=espana.py
    % Takes the maximum displacement value from the abaqus output file
    % "espana_displacement.rpt"
    !python getDisp_wing.py
    % Assigns the maximum displacement value to the variable "disp"
    fid = fopen('disp.dat', 'r');
```

```
disp = fscanf(fid, '%f', 1);
fclose(fid);
```

getDisp_wing.py

```
# Input  : espana_displacement.rpt
# Output : disp.dat
import re, os
a = open("espana_displacement.rpt", "r")
b = open("disp.dat", "w")
for i in a.readlines():
    disp = re.search(".*Maximum(.*)", i)
    if disp:
        yerd = float(disp.group(1))
        b.write(str(yerd))
a.close()
b.close()
os.remove("espana_displacement.rpt")
```

APPENDIX A.9

Reliability Based Aeroelastic Optimization of AGARD 445.6 Wing Matlab Code

nonlcon.m

```
function nonlcon()
% Output : output.dat
beta_stress();
beta_displacement();
exit
```

beta_stress.m

```
function beta_stress()
% This function evaluates the beta stress value for the given design
display('Beta Stress Calculation')
% R, Mach, Alfa
x_mean      = [8, 0.85, 5];          % Mean value
x_standard = [0.4, 0.03, 0.25];    % Standard deviation
h = 0.1;                                % For gradient calculation increment amount
convergence = 10;                     % To start the while loop, it is necessary
x = [x_mean(1), x_mean(2), x_mean(3)];  % Initial design point
u = [0, 0, 0];                          % Initial design point in the standard normal space
[row, col] = size(x);
i = 1;
while (convergence > 0.2)
   display('Iteration : ');
   grad = [];
   vonmises = vonmises_calculate(x)          % R, Mach, Alfa
   % cl,cd values are taken.
   if (i == 1)
      !python extract_cl_cd.py
      cl_cd = fopen('cl_cd.txt', 'r');
      a = fscanf(cl_cd,'%f');
      fclose(cl_cd);
```

```
  cl = a(1)
  cd = a(2)
end
m_g = x(1) - vonmises            % Limit state function's value at design point
% Gradient calculation using central finite differences method
for k = 1:col
  temp = x(k);
  x(k) = (u(k) + h) * x_standard(k) + x_mean(k)
  if (k ~= 1)
    vonmises = vonmises_calculate(x)
  end
  P2 = x(1) - vonmises
  x(k) = (u(k) - h) * x_standard(k) + x_mean(k)
  if (k ~= 1)
    vonmises = vonmises_calculate(x)
  end
  P1 = x(1) - vonmises
  grad(k) = ((P2 - P1) / (2 * h)) / x_standard(k)
  x(k) = temp;
end
% End gradient calculation
display('Gradient')
grad
sigma_g = sqrt(sum((grad .* x_standard).^2));
if i == 1
   display('MVFOSM Beta :')
   beta(i) = m_g / sigma_g
   alfa = - (grad .* x_standard) ./ sigma_g
else
   display('FORM Beta :')
   beta_upper = sum(grad .* x_standard .* u);
   beta(i) = (m_g - beta_upper) / sigma_g
   alfa = - (grad .* x_standard) ./ sigma_g
end
```

```
    display('New Design Point')
    x = x_mean + beta(i) .* x_standard .* alfa          % Compute a new design point.
    u = (x - x_mean) ./ x_standard
    % Check beta convergence
    if i == 1
       convergence = 10;
    else
       convergence = abs(beta(i) - beta(i-1)) / beta(i-1)
    end
    i = i + 1;
end
mpp   = [x(1), x(2), x(3)]                    % Most probable failure point (MPP)
converged_beta = beta(size(beta,2))         % Shortest distance to the MPP
f_write = fopen('output.dat', 'w');
% converged_beta, R, Mach, Alfa, cl, cd
fprintf(f_write,  '%f\n%f\n%f\n%f\n%f\n%f\n',  converged_beta,  mpp(1),  mpp(2),
mpp(3), cl, cd);
fclose(f_write);
```

vonmises_calculate.m

```
function vonmises = vonmises_calculate(x)
   yazdir_mach_alfa_1 = fopen('mach_alfa.txt', 'w');
   fprintf(yazdir_mach_alfa_1, '%f\n%f\n', x(2), x(3));
   fclose(yazdir_mach_alfa_1);
   !python fluent_change.py
   !fluent 3d -wait -i fluent.jou
   !xcopy /E "C:\Documents and Settings\aysan\Desktop\MnK-
Tez\Agard\template_mpcci" "mpcci_batch" /i
   !move agard.cas "mpcci_batch\fluent"
   !copy agard.inp "mpcci_batch\abaqus"
   cd  'mpcci_batch'
   !mpcci -batch mpcci_batch.csp
   !echo "mpcci finished"
   cd  ..
```

```
!move "mpcci_batch\fluent\cl-history"
!move "mpcci_batch\fluent\cd-history"
!move "mpcci_batch\abaqus\abaqus_run.odb"
!move "mpcci_batch\abaqus\abaqus_run.dat"
!rm -rf  "mpcci_batch"
!abq671.bat cae noGUI=abaqus.py
!python getMises.py
fid = fopen('mises.dat', 'r');
vonmises = fscanf(fid, '%f', 1);
fclose(fid);
```

fluent_change.py

```
import math, os
readed = open("mach_alfa.txt","r")
s = readed.readlines()
mach_new = float(s[0])
alfa = float(s[1])
readed.close()
a = open("fluent.jou", "r")
b = open("fluent_temp.jou","w")
alfa_cos = round(math.cos(math.radians(alfa)),3)
alfa_sin = round(math.sin(math.radians(alfa)),3)
k = 1
for i in a.readlines():
    if k == 23:
        b.write(str(mach_new) + "\n")
    elif k in [27,61,72]:
        b.write(str(alfa_cos) + "\n")
    elif k in [31,59,74]:
        b.write(str(alfa_sin) + "\n")
    else:
        b.write(i)
    k = k + 1
a.close()
```

```
b.close()
os.remove("fluent.jou")
os.rename("fluent_temp.jou", "fluent.jou")
```

getMises.py

```
# Input    : agard_mises.rpt
# Output : mises.dat
import re, os
a = open("agard_mises.rpt", "r")
b = open("mises.dat", "w")
for i in a.readlines():
    mises = re.search(".*Maximum(.*)", i)
    if mises:
        stress = float(mises.group(1))
        b.write(str(stress))
a.close()
b.close()
os.remove("agard_mises.rpt")
```

extract_cl_cd.py

```
a = open("cl-history","r")
f = open("cd-history","r")
cl_cd = open("cl_cd.txt","w")
for i in a.readlines():
    continue
b = i.split("\t")
c = float(b[1].strip())
for j in f.readlines():
    continue
ok = j.split("\t")
ok1 = float(ok[1].strip())
cl_cd.write(str(c) + "\n" + str(ok1))
cl_cd.close()
a.close()
```

```
f.close()
```

beta_displacement.m

```
function beta_displacement()
% This function evaluates the beta displacement value for the given design
display('Beta Displacement Calculation')
% Mach, Alfa
x_mean     = [0.85, 5];              % Mean value
x_standard = [0.03, 0.25];           % Standard deviation
h = 0.1;                             % For gradient calculation increment amount
convergence = 10;                    % To start the while loop, it is necessary
x = [x_mean(1), x_mean(2)]; % Initial design point
u = [0, 0];                          % Initial design point in the standard normal space
[row, col] = size(x);
i = 1;
D = 76.0;
while (convergence > 0.2)
   display('Iteration : ');
   grad = [];
   disp = disp_calculate(x)          % Mach, Alfa
   m_d = D - disp                    % Limit state function's value at design point
   % Gradient calculation using central finite differences method
   for k = 1:col
     temp = x(k);
     x(k) = (u(k) + h) * x_standard(k) + x_mean(k)
     disp = disp_calculate(x)
     P2 = D - disp
     x(k) = (u(k) - h) * x_standard(k) + x_mean(k)
     disp = disp_calculate(x)
     P1 = D - disp
     grad(k) = ((P2 - P1) / (2 * h)) / x_standard(k)
     x(k) = temp;
   end
   % End gradient calculation
```

```
    display('Gradient')
    grad
    sigma_g = sqrt(sum((grad .* x_standard).^2));
    if i == 1
        display('MVFOSM Beta :')
        beta(i) = m_d / sigma_g
        alfa = - (grad .* x_standard) ./ sigma_g
    else
        display('FORM Beta :')
        beta_upper = sum(grad .* x_standard .* u);
        beta(i) = (m_d - beta_upper) / sigma_g
        alfa = - (grad .* x_standard) ./ sigma_g
    end
    display('New Design Point')
    x = x_mean + beta(i) .* x_standard .* alfa          % Compute a new design point.
    u = (x - x_mean) ./ x_standard
    % Check beta convergence
    if i == 1
        convergence = 10;
    else
        convergence = abs(beta(i) - beta(i-1)) / beta(i-1)
    end
    i = i + 1;
end
mpp   = [x(1), x(2)]                        % Most probable failure point (MPP)
converged_beta = beta(size(beta,2))     % Shortest distance to the MPP
f_write = fopen('output.dat', 'a');
% converged_beta, Mach, Alfa
fprintf(f_write, '%f\n%f\n%f\n', converged_beta, mpp(1), mpp(2));
fclose(f_write);
```

disp_calculate.m

```
function disp = disp_calculate(x)
  yazdir_mach_alfa_1 = fopen('mach_alfa.txt', 'w');
```

```
   fprintf(yazdir_mach_alfa_1, '%f\n%f\n', x(1), x(2));
   fclose(yazdir_mach_alfa_1);
   !python fluent_change.py
   !fluent 3d -wait -i fluent.jou
   !xcopy /E "C:\Documents and Settings\aysan\Desktop\MnK-
Tez\agard_disp\template_mpcci" "mpcci_batch" /i
   !move agard.cas "mpcci_batch\fluent"
   !copy agard.inp "mpcci_batch\abaqus"
   cd  'mpcci_batch'
   !mpcci -batch mpcci_batch.csp
   !echo "mpcci finished"
   cd  ..
   !move "mpcci_batch\fluent\cl-history"
   !move "mpcci_batch\fluent\cd-history"
   !move "mpcci_batch\abaqus\abaqus_run.odb"
   !move "mpcci_batch\abaqus\abaqus_run.dat"
   !rm -rf  "mpcci_batch"
   !abq671.bat cae noGUI=abaqus.py
   % Displacement Calculation
   !python getDisp_wing.py
   fid = fopen('disp.dat', 'r');
   disp = fscanf(fid, '%f', 1);
   fclose(fid);
```

getDisp_wing.py

```
# Input    : agard_disp.rpt
# Output : disp.dat
import re, os
a = open("agard_disp.rpt", "r")
b = open("disp.dat", "w")
for i in a.readlines():
    disp = re.search(".*Maximum(.*)", i)
    if disp:
        yerd = float(disp.group(1))
```

```
        b.write(str(yerd) + "\n")
        break
a.close()
b.close()
os.remove("agard_disp.rpt")
```

CURRICULUM VITAE

Candidate's full name: Muhammet Nasıf KURU

Place and date of birth: Mersin, 1986

Permanent Address: Anıt Mah. 0331 Sok. Tellioğlu Apt. 39/5 Tarsus - Mersin

Universities and Colleges attended: Dokuz Eylul University, Mechanical Engineering, 2007

Publications:

- M. Nikbay, **M. N. Kuru**, S. Bayat and N. Fakkusoglu, 2011: Reliability-Based Design Optimisation of an Aircraft Wing Structure with High-Fidelity Modelling. *International Journal of Design Engineering*, Vol. 3, No. 3, pp 247-259.

- M. Nikbay, N. Fakkusoğlu, and **M. N. Kuru**, 2010: Reliability Based Multi-disciplinary Optimization of Aeroelastic Systems with Structural and Aerodynamic Uncertainties. *13th AIAA/ISSMO, Multidisciplinary Analysis and Optimization (MAO) Conference*, 13-15 September 2010, Fort Worth, Texas, USA.

- M. Nikbay, N. Fakkusoğlu, and **M. N. Kuru**, 2010: Reliability Based Aeroelastic Optimization of a Composite Aircraft Wing via Fluid-Structure Interaction of High Fidelity Solvers. *WCCM/APCOM2010 9th World Congress on Computational Mechanics and 4th Asian Pacific Congress on Computational Mechanics*, 19-23 July, 2010, Sydney, Australia.

- M. Nikbay, **M. N. Kuru**, S. Bayat and N. Fakkusoglu, 2010: Implementation of a Reliability Based Design Algorithm for High-Fidelity Structural Optimization of an Aircraft Wing. *14.International Conference on Machine Design and Production*, June 29 - July 2, 2010, Güzelyurt, Northern Cyprus.

- M. Nikbay, N. Fakkusoğlu and **M. N. Kuru**, 2010: Reliability Based Structural and Aeroelastic Optimization of Aircraft Wing with Multidisciplinary Code Coupling. *ASMDO Third International Conference on Multidisciplinary Design Optimization and Applications*, 21-23 June 2010**,** Paris, France.